CHAPTER

7

Test synthesis

Laung-Terng (L.-T.) Wang
SynTest Technologies, Inc., Sunnyvale, California

Xiaoqing Wen
Kyushu Institute of Technology, Fukuoka, Japan

Shianling Wu
SynTest Technologies, Inc., Princeton Junction, New Jersey

ABOUT THIS CHAPTER

Test synthesis is an important step in VLSI testing for automating the process of producing testable VLSI designs. The test synthesis flow typically includes testability rule checking and repair in the beginning to guarantee that the design has complied with all given testability rules. Once all rules are met, test synthesis is then performed to automatically insert test logic into the design. The test logic can include ***design-for-testability*** (DFT) circuitry used for scan design, logic ***built-in self-test*** (BIST), and/or test compression. Depending on the test requirements, additional circuitries for ***design-for-debug-and-diagnosis*** (DFD) and ***design-for-reliability*** (DFR) could also be inserted. These circuitries altogether are intended for improving the quality and reducing the test cost of the manufactured devices as well as simplifying the test, debug, and diagnosis tasks.

The focus of this chapter is on widely used scan synthesis flow and an emerging BIST synthesis flow. A set of scan design rules used in the scan synthesis flow, with which a design must comply, is described first. This is followed by the gate-level scan synthesis flow. Discussion then moves to new design rules required for logic BIST in the emerging BIST synthesis flow. This is followed by a BIST design example with all necessary steps involved in designing the logic BIST system, verifying its correctness, and further improving its fault coverage. The chapter concludes with a discussion of the scan design flow at the ***register-transfer level*** (RTL) which helps further reduce DFT design iterations and test development time. The RTL scan design flow can be readily extended for logic synthesis and integrated with other advanced DFT, DFD, and DFR features implemented at the RTL.

7.1 INTRODUCTION

Test synthesis is a **test automation** process to audit whether a digital design complies with all **testability rules** and then insert the required test logic into the digital design. In modern test synthesis tools, the automatic repair function, which automatically modifies the digital design to make it comply with the **testability rules**, is often provided. The digital design can be specified at the gate level or at the *register-transfer level* (RTL). After all testability rule violations are identified and repaired either manually or automatically, the resulting digital design is often referred to as a **testable design** or **test-ready core**.

By using the scan method, the test logic to be incorporated into the testable design will require conversion of internal storage elements, such as D flip-flops or D latches, into **scan cells**, such as *muxed-D scan cells* or *LSSD scan cells*. The resulting testable design is often called a **scan design**. In addition to being the dominant DFT architecture used for detecting manufacturing defects, scan design has become the backbone for more advanced DFT techniques, such as logic BIST and test compression [Wang 2006a]. Furthermore, as designs continue to move toward the nanometer scale, scan design is being used as a design feature to facilitate silicon debug, fault diagnosis, failure analysis, and reliability enhancement against **soft errors** [SIA 2005, 2006; Gizopoulos 2006; Wang 2007].

Recently, ***design for testability*** (DFT) has started to migrate from the gate level to the RTL. The motivation for this migration is to allow integration of additional DFT features, such as logic BIST and test compression, at the RTL to reduce test development time and to create reusable and testable RTL cores. This further allows the integrated RTL DFT design to be processed as part of synthesis-based optimization as to reduce test power, performance degradation, and area overhead.

Figure 7.1 shows a typical logic BIST system that has been synthesized in a scan design with embedded logic BIST and test compression circuitry. The logic BIST circuitry is composed of a logic BIST controller, a ***test pattern generator*** (TPG), and an ***output response analyzer*** (ORA). The test compression circuitry includes a **decompressor** and a **compactor**. Depending on the nature of the ***circuit under test*** (CUT) or scan design, the decompressor and the compactor can be a part of the TPG and the ORA, respectively. However, for the purpose of illustration, we separate the logic BIST circuitry from the test compression circuitry. Generally speaking, the logic BIST system will, at the minimum configuration, operate in five modes: normal mode, logic BIST mode, ATPG compression mode, ATPG mode, and serial debug and diagnosis mode.

In normal mode, all test logic will be disabled, and the CUT will function as intended. When the logic BIST system operates in BIST mode, the TPG will bypass the decompressor and automatically generate test patterns for application to the CUT (refer to Figure 3.13 in Chapter 3). The compactor will be bypassed, and the ORA will automatically compact the output responses of the CUT into a *signature*. Specific BIST timing control signals, including scan enable signals and clocks, are generated by the logic BIST controller for

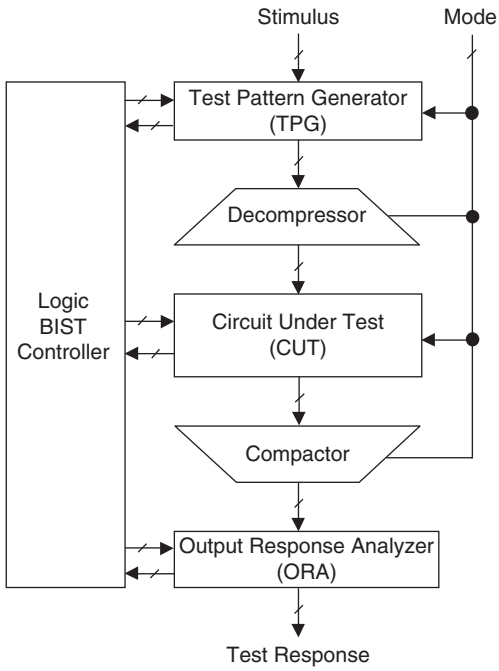


FIGURE 7.1

A typical logic BIST system with test compression circuitry.

coordinating the BIST operation among the TPG, CUT, and ORA. The logic BIST controller provides a pass/fail indication once the BIST operation is complete. It includes comparison logic to compare the *final signature* with an embedded expected or *golden signature*, and often comprises diagnostic logic for fault diagnosis. Because the ORA usually cannot tolerate unknown (X) values during output response analysis, it is required that all storage elements in the TPG, CUT, and ORA be initialized to known states before self-test, and no unknown (X) values be allowed to propagate from the CUT to the ORA. In other words, the CUT must comply with additional **BIST-specific design rules**.

When the system operates in ATPG compression mode, the TPG will be bypassed, and the decompressor will take over and decompress external compressed scan patterns (compressed stimuli) from *automatic test equipment* (ATE) for application to the scan data inputs of the CUT. At the same time, the compactor will compact the scan data outputs of the CUT, bypass the ORA, and shift the compressed test responses out to the ATE for comparison.

When the system operates in ATPG mode, both logic BIST and test compression circuitry will be bypassed. Scan patterns (stimuli) are directly shifted in from the ATE to the CUT, and the test responses are immediately shifted out to the ATE for comparison. In contrast to the ATPG mode, the serial debug and diagnosis mode requires that all scan chains be concatenated into one serial scan chain so

test responses from the CUT can be shifted out bit-by-bit to a target computer or the ATE for analysis. This mode of operation is especially important for on-board or in-system diagnosis when the chip operates in the field.

7.2 SCAN DESIGN

Scan design is currently the most widely used structured DFT approach. As discussed in Chapter 3, it is implemented by connecting selected storage elements of a design into one or more shift registers, called **scan chains**, to provide external access to the storage elements. Scan design is obtained by replacing all selected storage elements with scan cells, each scan cell having one additional ***scan input*** (SI) port and one shared or additional ***scan output*** (SO) port. By connecting the SO port of one scan cell to the SI port of the next scan cell, one or more scan chains are created.

For a scan design to achieve the desired ***defective parts per million*** (PPM) goal, specific circuit structures and design practices that may affect fault coverage must be identified and repaired. This requires compiling a set of **scan design rules** that the design must adhere to. Hence, it is required to identify and fix scan design rule violations in the design before inserting or synthesizing scan chains into the design and generating test patterns for the scan design.

7.2.1 Scan design rules

To implement scan into a design, the design must comply with a set of scan design rules [Cheung 1997]. In addition, certain design styles must be avoided, because they may limit the fault coverage that can be achieved. A number of scan design rules that are required to successfully use scan and achieve the target fault coverage goal are listed in Table 7.1. In this table, a possible solution is recommended for each scan design rule violation. Scan design rules that are labeled "avoid" must be repaired throughout the shift and capture operations. Scan design rules that are labeled "avoid during the shift" must be fixed only during the shift operation. Detailed descriptions are provided for some critical scan design rules.

7.2.1.1 *Tristate buses*

Bus contention occurs when two bus drivers force opposite logic values onto a tristate bus, which can damage the chip. Bus contention is designed not to happen during the normal operation and is typically avoided during the capture operation, because advanced ATPG programs can generate test patterns that guarantee only one bus driver controls a bus. However, during the shift operation, no such guarantees can be made; therefore, certain modifications must be made to each tristate bus to ensure that only one driver controls the bus. For example, for the tristate bus shown in Figure 7.2a, which has three bus drivers (D_1, D_2, and D_3), circuit modification can be made as shown in Figure 7.2b,

Table 7.1 Typical Scan Design Rules

Design Style	Scan Design Rule	Recommended Solution
Tristate buses	Avoid during shift	Fix bus contention during shift
Bidirectional I/O ports	Avoid during shift	Force to input or output mode during shift
Gated clocks (muxed-D full-scan)	Avoid during shift	Enable clocks during shift
Derived clocks (muxed-D full-scan)	Avoid	Bypass clocks
Combinational feedback loops	Avoid	Break the loops
Asynchronous set/ reset signals	Avoid	Use external pin(s)
Clocks driving data	Avoid	Block clocks to the data portion
Floating buses	Avoid	Add bus keepers
Floating inputs	Not recommended	Tie to V_{DD} or V_{SS}
Cross-coupled NAND/NOR gates	Not recommended	Use standard cells
Non-scan storage elements	Not recommended for full-scan design	Initialize to known states, bypass, or make transparent

where EN_1 is forced to 1 to enable the D_1 bus driver, whereas EN_2 and EN_3 are set to 0 to disable both D_2 and D_3 bus drivers, when $SE = 1$.

In addition to bus contention, a bus without a pull-up, pull-down, or bus keeper may result in fault coverage loss. The reason is that the value of a floating bus is unpredictable, which makes it difficult to test for a stuck-at-1 fault at the enable signal of a bus driver. To solve this problem, a pull-up, pull-down, or bus keeper can be added. The bus keeper added in Figure 7.2b is an example of fixing this problem by forcing the bus to preserve the logic value driven onto it prior to when the bus becomes floating.

7.2.1.2 *Bidirectional I/O ports*

Bidirectional I/O ports are used in many designs to increase the data transfer bandwidth. During the capture operation, a bidirectional I/O port is usually specified as being either input or output; however, conflicts may occur at a bidirectional I/O port during the shift operation. An example is shown in Figure 7.3a, in which a bidirectional I/O port is used as an input, and the direction control is provided by the scan cell. Because the output value of the scan cell can vary

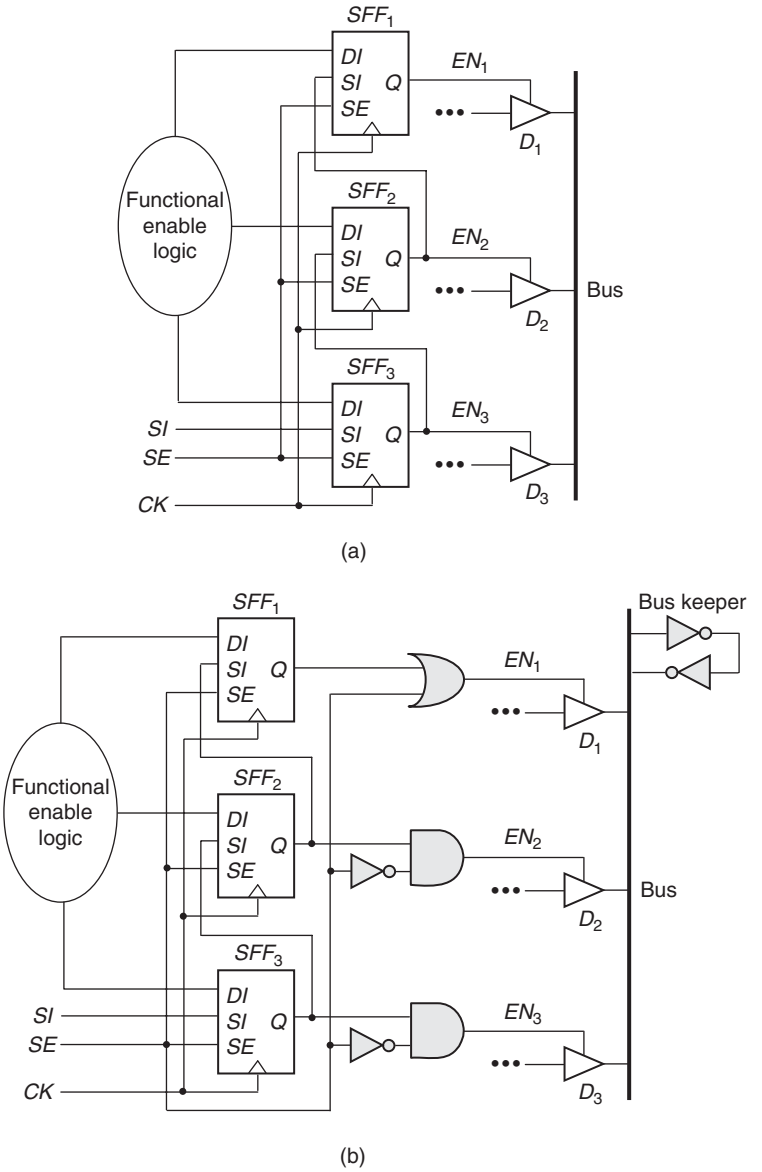


FIGURE 7.2

Fixing bus contention: (a) Original circuit. (b) Modified circuit.

during the shift operation, the output tristate buffer may become active, resulting in a conflict if *BO* and the I/O port driven by the tester have opposite logic values. Figure 7.3b shows an example of how to fix this problem by forcing the tristate buffer to be inactive when $SE = 1$, and the tester is used to drive

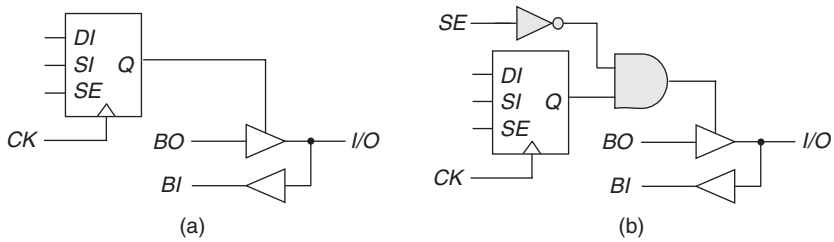


FIGURE 7.3

Fixing bidirectional I/O ports: (a) Original circuit. (b) Modified circuit.

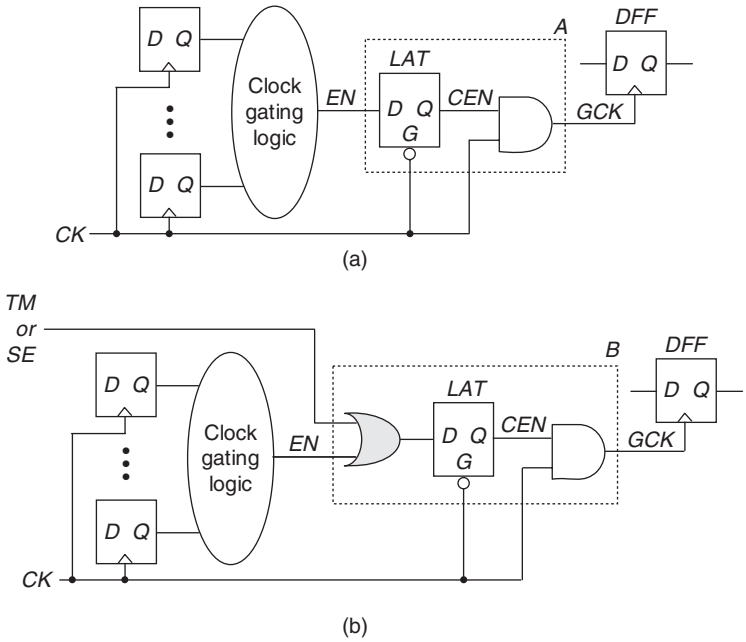


FIGURE 7.4

Fixing gated clocks: (a) Original circuit. (b) Modified circuit.

the I/O port during the shift operation. During the capture operation, the applied test vector determines whether a bidirectional I/O port is used as input or output and controls the tester appropriately.

7.2.1.3 *Gated clocks*

Clock gating is a widely used design technique for reducing power by eliminating unnecessary switching activity at storage elements. An example is shown in Figure 7.4a. The clock enable signal (*EN*) is generated at the rising edge of *CK*

and is loaded into the latch *LAT* at the failing edge of *CK* to become *CEN*. *CEN* is then used to enable or disable clocking for the flip-flop *DFF*. Although clock gating is a good approach for reducing power consumption, it prevents the clock ports of some flip-flops from being directly controlled by primary inputs. As a result, modifications are necessary to allow the scan shift operation to be conducted on these storage elements.

The clock gating function should be disabled, at least during the shift operation. Figure 7.4b shows how the clock gating can be disabled. In this example, an OR gate is used to force *CEN* to 1 with either the test mode signal *TM* or the scan enable signal *SE*. If *TM* is used, *CEN* will be held at 1 during the entire scan test operation (including the capture operation). This will make it impossible to detect faults in the clock gating logic, causing fault coverage loss. If *SE* is used, *CEN* will be held at 1 only during the shift operation but will be released during the capture operation; hence, higher fault coverage can be achieved but at the expense of increased test generation complexity.

7.2.1.4 *Derived clocks*

A derived clock is a clock signal generated internally from a storage element or a clock generator, such as ***phase-locked loop*** (PLL), frequency divider, or pulse generator. Because derived clocks are not directly controllable from primary inputs, to test the logic driven by these derived clocks, these clock signals must be bypassed during the entire test operation. An example is illustrated in Figure 7.5a, in which the derived clock *ICK* drives the flip-flops DFF_1 and DFF_2. In Figure 7.5b, a multiplexer selects *CK*, which is a clock directly controllable from a primary input, to drive DFF_1 and DFF_2 during the entire test operation when $TM = 1$.

7.2.1.5 *Combinational feedback loops*

Depending on whether the number of inversions on a combinational feedback loop is even or odd, it can introduce either sequential behavior or oscillation into a design. Because the value stored in the loop cannot be controlled or determined

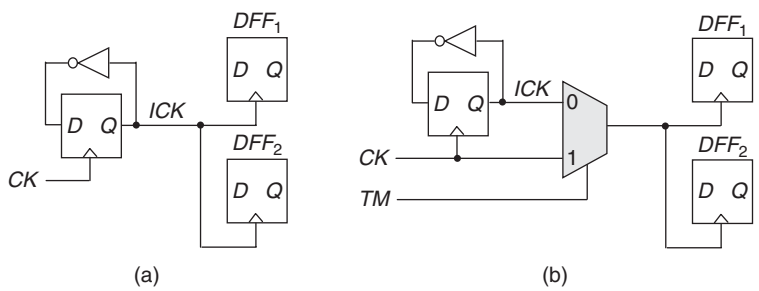


FIGURE 7.5

Fixing derived clocks: (a) Original circuit. (b) Modified circuit.

during test, this can lead to an increase in test generation complexity or fault coverage loss. Because combinational feedback loops are not a recommended design practice, the best way to fix this problem is to rewrite the RTL code generating the loop. In cases where this is not possible, a combinational feedback loop, as shown in Figure 7.6a, can be fixed by using a test mode signal *TM*. This signal permanently disables the loop throughout the entire shift and capture operations by inserting a scan point (*i.e.*, a combination of control and observation points) to break the loop, as shown in Figure 7.6b.

7.2.1.6 *Asynchronous set/reset signals*

Asynchronous set/reset signals of scan cells that are not directly controlled from primary inputs can prevent scan chains from shifting data properly. To avoid this problem, it is required that these asynchronous set/reset signals be forced to an inactive state during the shift operation. These asynchronous set/reset signals are typically referred to as being sequentially controlled. An example of a sequentially controlled reset signal *RL* is shown in Figure 7.7a. A method for

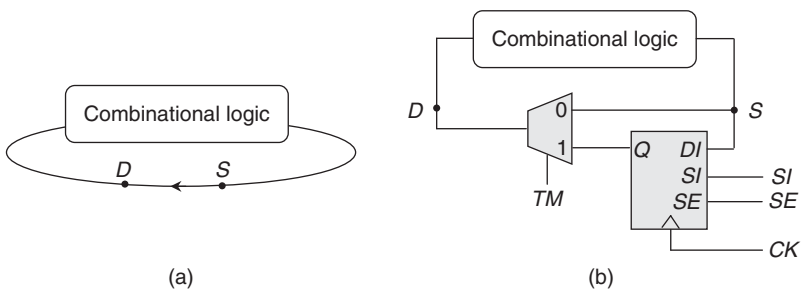


FIGURE 7.6

Fixing combinational feedback loops: (a) Original circuit. (b) Modified circuit.

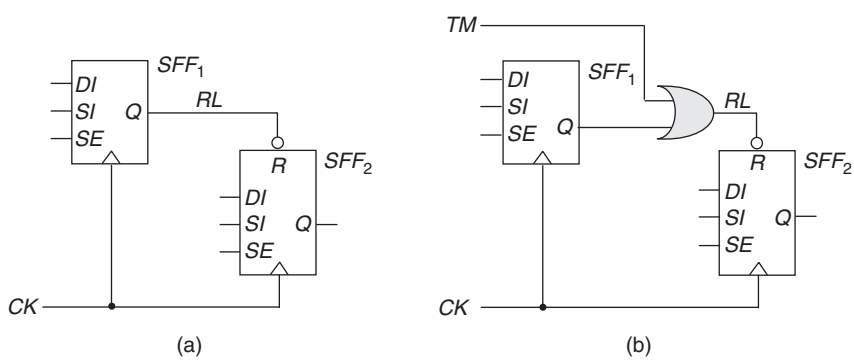


FIGURE 7.7

Fixing combinational feedback loops: (a) Original circuit. (b) Modified circuit.

fixing this asynchronous reset problem with an OR gate with an input tied to the test mode signal *TM* is shown in Figure 7.7b. When $TM = 1$, the asynchronous reset signal *RL* of scan cell SFF_2 is permanently disabled during the entire test operation.

The disadvantage of using the test mode signal *TM* to disable asynchronous set/reset signals is that faults within the asynchronous set/reset logic cannot be tested. The use of the scan enable signal *SE* instead of *TM* makes it possible to detect faults within the asynchronous set/reset logic, because during the capture operation ($SE = 0$), these asynchronous set/reset signals are not forced to the inactive state. However, this might result in mismatches because of race conditions between the clock and asynchronous set/reset ports of the scan cells. A better solution is to use an independent reset enable signal *RE* to replace *TM* and to conduct test generation in two phases. In the first phase, *RE* is set to 1 during both shift and capture operations to test data faults through the *DI* port of the scan cells while all asynchronous set/reset signals are held inactive. In the second phase, *RE* is set to 1 during the shift operation and 0 during the capture operation without applying any clocks to test faults within the asynchronous set/reset logic.

7.2.2 Scan design flow

Although conceptually scan design is not difficult to understand, the practice of inserting scan into a design to turn it into a scan design requires careful planning. This often requires many circuit modifications for which care must be taken not to disrupt the normal functionality of the circuit. In addition, many physical implementation details must be taken into consideration to guarantee that scan testing can be performed successfully. Finally, a good understanding of scan design, with respect to which scan cell design and scan architecture to use, is required to better plan in advance which scan design rules must be complied with and which debug and diagnose features must be included to facilitate simulation, debug, and fault diagnosis [Gizopoulos 2006; Wang 2007].

The shift operation and the capture operation are the two key scan operations in which care needs to be taken to guarantee that the scan design can operate properly. The shift operation, which is common to all scan designs, must be designed to perform successfully, regardless of the clock skew that exists within the same clock domain and between different clock domains. The capture operation is also common to all scan designs, albeit with more stringent scan design rules in some scan designs compared with others. It must be designed such that the ATPG tool is able to correctly and deterministically predict the expected responses of the generated test patterns. This requires a basic understanding of the logic simulation and fault models used for ATPG, as well as the clocking scheme used during the capture operation.

A typical design flow for implementing scan in a sequential circuit is shown in Figure 7.8. In this figure, scan design rule checking and repair are first

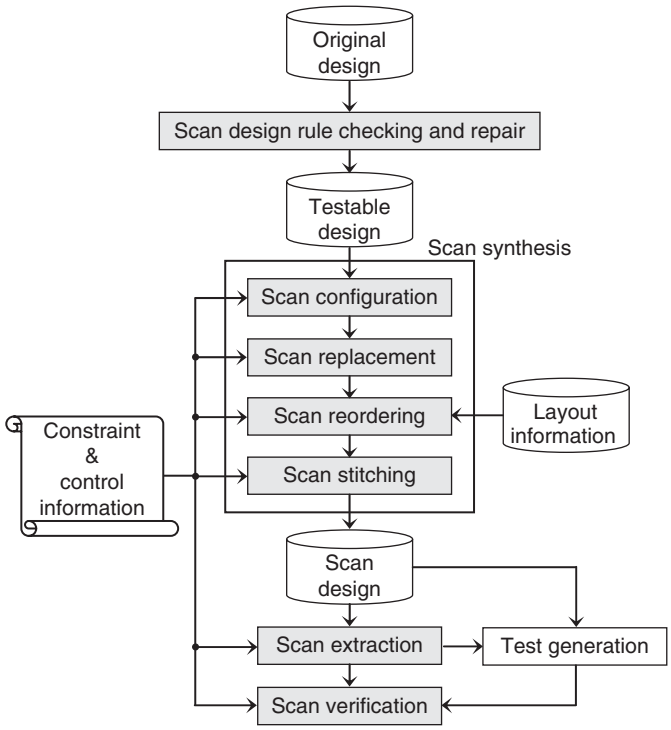


FIGURE 7.8

Typical scan design flow.

performed on a pre-synthesis RTL design or on a post-synthesis gate-level design, typically referred to as a **netlist**. The resulting design after scan repair is referred to as a **testable design**. Once all scan design rule violations are identified and repaired, scan synthesis is performed to convert the testable design into a scan design. The scan design now includes one or more scan chains for scan testing. A scan extraction step is used to further verify the integrity of the scan chains and to extract the final scan architecture of the scan chains for ATPG. Finally, scan verification is performed on both shift and capture operations to verify that the expected responses predicted by the zero-delay simulator used in test generation or fault simulation match with the full-timing behavior of the circuit under test. The steps shown in the scan design flow are described in the following subsections in more detail.

7.2.2.1 *Scan design rule checking and repair*

The first step in implementing a scan design is to identify and repair all scan design rule violations to convert the original design into a testable design.

Repairing these violations allows the testable design to meet target fault coverage requirements and guarantees that the scan design will operate correctly. These scan design rules were described in the previous section. In addition to these scan design rules, certain clock control structures may have to be added for at-speed delay testing. Typically, scan design rule checking is also performed on the scan design after scan synthesis to confirm that no new violations occur.

On successful completion of this step, the testable design must guarantee the correct shift and capture operations. During the shift operation, all clocks controlling scan cells of the design are directly controllable from external pins. The clock skew between adjacent scan cells must be properly managed so as to avoid any shift failure. During the capture operation, fixing all scan design rule violations should guarantee correctness for data paths that originate and terminate within the same clock domain. For data paths that originate and terminate in different clock domains, additional care must be taken in terms of the way the clocks are applied to guarantee the success of the capture operation. This is mainly because the clock skew between different clock domains is typically large. A data path originating in one clock domain and terminating in another might result in a mismatch when both clocks are applied simultaneously, and the clock skew between the two clocks is larger than the data path delay from the originating clock domain to the terminating clock domain. To avoid the mismatch, the timing governing the relationship of such a data path shown in the following equation must be observed:

$$\textit{Clock skew} < \textit{Data path delay} + \textit{Clock-to-Q delay}\ (\textit{originating clock})$$

If this is not the case, a mismatch may occur during the capture operation. To prevent this from happening, clocks belonging to different clock domains can be applied sequentially (with the **staggered clocking** scheme), as opposed to simultaneously, such that any clock skew that exists between the clock domains can be tolerated during the test generation process. It is also possible to apply only one clock during each capture operation with the **one-hot clocking** scheme. On the other hand, a design typically contains a number of noninteracting clock domains. In this case, these clocks can be applied simultaneously, which can reduce the complexity and the final pattern count of the pattern generation and fault simulation process. **Clock grouping** is a process used to identify all independent or noninteracting clocks that can be grouped and applied simultaneously.

An example of the clock grouping process is shown in Figure 7.9. This example shows the results of performing a circuit analysis operation on a testable design to identify all clock interactions, marked with an arrow, where a data transfer from one clock domain to a different clock domain occurs. As seen in Figure 7.9, the circuit in this example has seven clock domains ($CD_1 \sim CD_7$) and five cross-clock-domain data paths ($CCD_1 \sim CCD_5$). From this example, it can be seen that CD_2 and CD_3 are independent from each other; hence, their related clocks can be applied simultaneously during test as CK_2. Similarly, clock

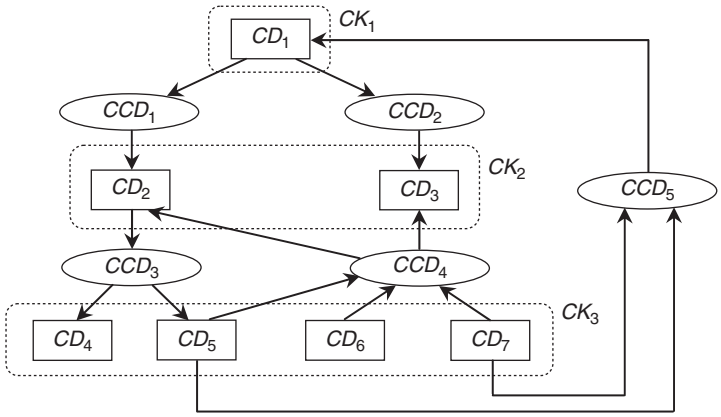


FIGURE 7.9

Clock grouping example.

domains CD_4 through CD_7 can also be applied simultaneously during test as CK_3. Therefore, in this example, three grouped clocks instead of seven individual clocks can be used to test the circuit during the capture operation.

7.2.2.2 *Scan synthesis*

When all the repairs of scan design rule violations have been made to the circuit, the scan synthesis flow is commenced. The scan synthesis flow converts a testable design into a scan design without affecting the functionality of the original design. Static analysis tools and equivalence checkers, which compare the logic circuitry of two circuits under given constraints, are typically used to verify that this is, indeed, the case. Depending on the types of scan cells used and the types of scan architecture implemented, minor modifications to the scan synthesis flow shown in Figure 7.8 may be necessary.

During the 1990s, this scan synthesis operation was typically performed with a separate set of scan synthesis tools, which were applied after the logic synthesis tool had synthesized a gate-level netlist out of an RTL description of the design. Recently, these scan synthesis features are being integrated into the logic synthesis tools, and scan designs are synthesized automatically from the RTL. The process of performing scan synthesis during logic synthesis is often referred to as **one-pass synthesis** or **single-pass synthesis**.

The scan synthesis flow shown in Figure 7.8 includes four separate steps: (1) scan configuration, (2) scan replacement, (3) scan reordering, and (4) scan stitching. Each of these steps is described in the following in more detail.

7.2.2.2.1 Scan configuration

Scan configuration describes the initial step in scan chain planning, in which the general structure of the scan design is determined. The main decisions that

are made at this stage include: (1) the number of scan chains used; (2) the types of scan cells used to implement these scan chains; (3) storage elements to be excluded from the scan synthesis process; and (4) the way the scan cells are arranged within the scan chains.

The number of scan chains used is typically determined by analyzing the input and output pins of the circuit to determine how many pins can be allocated for the scan use. So as not to increase the number of pins of the circuit, which is typically limited by the size of the die, scan inputs and outputs are shared with existing pins during scan testing. In general, the larger the number of scan chains used, the shorter the time to perform a test on the circuit. This is because the maximum length of the scan chains dictates the overall test application time required to run each test pattern. One limitation that can preclude many scan chains from being used is the presence of high-speed I/O pads. The addition of any wire load to the high-speed I/O pads may adversely affect the timing of the design. An additional limitation is the number of tester channels available for scan testing.

The second issue regarding the types of scan cells to use typically depends on the process library. In general, for each type of storage element used, most process libraries have a corresponding scan cell type that closely resembles the functionality and timing of the storage element during the normal operation.

The third issue relates to which storage elements to exclude from scan synthesis. This is typically determined by investigating parts of the design where replacing storage elements with functionally equivalent scan cells may adversely affect timing. Therefore, storage elements lying on the critical paths of a design where the timing margin is very tight are often excluded from the scan replacement step to guarantee that the manufactured device will meet the restricted timing. In addition, certain parts of a design may be excluded from the scan for many different reasons, including security reasons (*e.g.*, parts of a circuit that deal with encryption). In these cases, individual storage element types, individual storage element instances, or a complete section of the design can be specified as "don't scan."

The remaining issue is to determine how the storage elements are arranged within the scan chains. This typically depends on how the number of clock domains relates to the number of scan chains in the design. In general, a scan chain is formed out of scan cells belonging to a single clock domain. For clock domains that contain a large number of scan cells, several scan chains are constructed, and a scan-chain balancing operation is performed on the clock domain to reduce the maximum scan-chain length. Oftentimes, a clock domain will include both negative-edge and positive-edge scan cells. If the number of negative-edge scan cells in a clock domain is large enough to construct a separate scan chain, these scan cells can be allocated as such. In cases where a scan chain has to include both negative-edge and positive-edge scan cells, all negative-edge scan cells are arranged in the scan chains such that they precede all

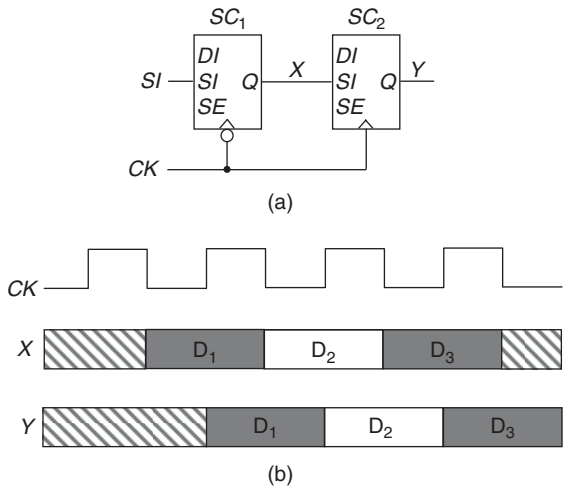


FIGURE 7.10

Mixing negative-edge and positive-edge scan cells in a scan chain: (a) Circuit structure. (b) Timing diagram.

positive-edge scan cells to guarantee that the shift operation can be performed correctly.

Figure 7.10a shows an example of a circuit structure made up of a negative-edge scan cell followed by a positive-edge scan cell. The associated timing diagram, shown in Figure 7.10b, illustrates the correct shift timing of the circuit structure. During each shift clock cycle, Y will first take on the state X at the rising CK edge before X is loaded with the SI value at the falling CK edge. If we accidentally place the positive-edge scan cell before the negative-edge scan cell, both scan cells will always incorrectly contain the same value at the end of each shift clock cycle.

In cases where scan chains must include scan cells from several different clock domains, a lock-up latch is inserted between adjacent cross-clock-domain scan cells to guarantee that any clock skew between the clocks can be tolerated. Clock skew between different clock domains is expected, because clock skew is controlled within a clock domain to remain below a certain threshold but not controlled across different clock domains. As a result, a race caused by a hold time violation could occur between these two scan cells if a lock-up latch is not inserted.

Figure 7.11a shows an example of a circuit structure having a scan cell SC_p belonging to clock domain CK_1 driving a scan cell SC_q belonging to clock domain CK_2 through a lock-up latch. The associated timing diagram is shown in Figure 7.11b, where CK_2 arrives after CK_1, to demonstrate the effect of clock skew on cross-clock-domain scan cells. During each shift clock cycle, X will first take on the SI value at the rising CK_1 edge, then Z will take on the Y value at

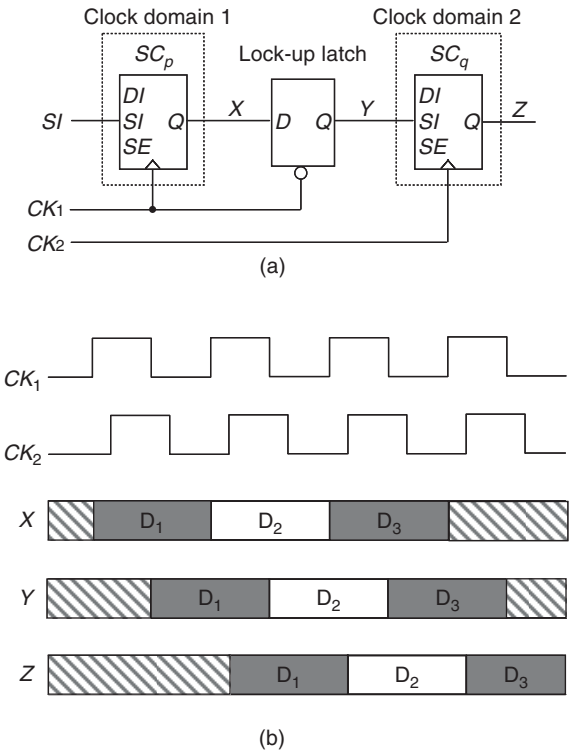


FIGURE 7.11

Adding a lock-up latch between cross-clock-domain scan cells: (a) Circuit structure. (b) Timing diagram.

the rising CK_2 edge. Finally, the new X value is transferred to Y at the falling CK_1 edge to store the SC_p contents. If CK_2 arrives earlier than CK_1, Z will first take on the Y value at the rising CK_2 edge. Then, X will take on the SI value at the rising CK_1 edge. Finally, the new X value is transferred to Y at the falling CK_1 edge to store the SC_p contents. In both cases, the lock-up latch design in Figure 7.11a allows correct shift operation regardless of whether CK_2 arrives earlier or later than CK_1. It is important to note that this scheme works only when the clock skew between CK_1 and CK_2 is less than the width (duty cycle) of the clock pulse. If this is not the case, then slowing down the shift clock frequency or enlarging the duty cycle of the shift clock can guarantee that this approach will work for any amount of clock skew. Other lock-up latch and lock-up flip-flop designs can also be used.

Once the clock structure of the scan chains is determined, it is still necessary to determine which scan cells should be stitched together into one scan chain and the order in which these scan cells should be placed. In some scan

synthesis flows, a preliminary layout placement is used to allocate scan cells to different scan chains belonging to the same clock domain. Then, the best order in which to stitch these scan cells within the scan chains is determined to minimize the scan routing required to connect the output of each scan cell to the scan input of the next scan cell. In cases where a preliminary placement is not available, scan cells can be assigned to different scan chains on the basis of an initial floorplan of the testable design by grouping scan cells in proximate regions of the design together. Once the final placement is determined, the scan chains can then be reordered and stitched, and the scan design is modified on the basis of the new scan chain order.

7.2.2.2.2 Scan replacement

After scan configuration is complete, **scan replacement** replaces all original storage elements in the testable design with their functionally equivalent scan cells. The testable design after scan replacement is often referred to as a **scan-ready design**. Functionally equivalent scan cells are the scan cells that most closely match power, speed, and area characteristics of the original storage elements. The scan inputs of these scan cells are often tied to the scan outputs of the same scan cell to prevent floating inputs from being present in the circuit. These connections are later removed during the scan-stitching step. In cases where one-pass or single-pass synthesis is used, scan replacement is transparent to tool users. Recently, some RTL scan-synthesis tools have implemented scan replacement at the RTL, even before going to the logic/scan synthesis tool, to reflect the scan design changes in the original RTL design.

7.2.2.2.3 Scan reordering

Scan reordering refers to the process of reordering scan cells in scan chains on the basis of the physical scan cell locations to minimize the number of interconnect wires used to implement the scan chains. During design implementation, if the physical location of each scan cell instance is not available, a "random" scan order based purely on the module-level and bus-level connectivity of the testable design can be used. However, if a preliminary placement is available, scan cells can be assigned to different scan chains on the basis of the initial floorplan of the design. Only after the final placement process of the physical implementation is performed on this testable design is the physical location of each scan cell instance taken into consideration. During the routing process of the physical implementation, scan reordering can be performed with *intrascan-chain reordering*, *interscan-chain reordering*, or a combination of both. **Intrascan-chain reordering**, in which scan cells are reordered only within their respective scan chains, does not reorder any scan cells across clock or clock-polarity boundaries. **Interscan-chain reordering**, in which scan cells are reordered among different scan chains, must make sure that the clock structure of the scan chains is preserved. In both intrascan-chain reordering and interscan-chain reordering, care must also be taken to limit the minimum

distance between scan cells to avoid timing violations that can destroy the integrity of the shift operation.

Advanced techniques have also been proposed to further reduce routing congestion while avoiding timing violations during the shift operation [Duggirala 2002, 2004]. For deep submicron circuits, the capacitance of the scan chain interconnect must also be taken into account to guarantee correct shift operation [Barbagallo 1996].

7.2.2.2.4 Scan stitching

Finally, the **scan-stitching** step is performed to stitch all scan cells together to form scan chains. Scan stitching refers to the process of connecting the output of each scan cell to the scan input of the next scan cell on the basis of the scan order specified previously. An additional step is also performed by connecting the scan input of the first scan cell of each scan chain to the appropriate scan chain input port and the scan output of the last scan cell of each scan chain to the appropriate scan chain output port to make the scan chains externally accessible. In cases where a shared I/O port is used to connect to the scan chain input or the scan chain output, additional signals must be connected to the shared I/O port to guarantee that it always behaves as either input or output, respectively, throughout the shift operation. As mentioned earlier, it is important to avoid the use of high-speed I/O ports as scan chain inputs or outputs, because the additional loading could result in a degradation of the maximum speed at which the device can be operated. In addition to stitching the existing scan cells, lock-up latches or lock-up flip-flops are often inserted during the scan-stitching step for adjacent scan cells where clock skew may occur. These lock-up latches or lock-up flip-flops are then stitched between adjacent scan cells.

7.2.2.3 *Scan extraction*

When the scan stitching step is complete, the scan synthesis process is complete. The original design has now been converted into a scan design; however, an additional step is often performed to verify the integrity of the scan chains, especially if any design changes are made to the scan design. **Scan extraction** is the process used for extracting all scan cell instances from all scan chains specified in the scan design. This procedure is performed by tracing the design for each scan chain to verify that all the connections are intact when the design is placed in shift mode. Scan extraction can also be used to prepare for the test generation process to identify the scan architecture of the design in cases where this information is not otherwise available.

7.2.2.4 *Scan verification*

When the physical implementation of the scan design is completed, including placement and routing of all the cells of the design, a timing file in ***standard delay format*** (SDF) is generated. This timing file resembles the timing

behavior of the manufactured device. This is then used to verify that scan testing can be successfully performed on the manufactured scan design.

Other than the trivial problems of scan chains being incorrectly stitched, verification errors during the shift operation are typical because of hold time violations between adjacent scan cells, where the data path delay from the output of a driving scan cell to the scan input of the following scan cell is smaller than the clock skew that exists between the clocks driving the two scan cells. In cases where the two scan cells are driven by the same clock, this may indicate a failure of the ***clock tree synthesis*** (CTS) process in guaranteeing that the clock skew between scan cells belonging to the same clock domain be kept at a minimum. In cases where the two scan cells are driven by different clocks, this may indicate a failure of inserting a required lock-up latch between the scan cells of the two different clock domains.

Apart from clock skew problems, other scan shift problems may occur. Often, they stem from (1) an incorrect scan initialization sequence that fails to put the design into test mode; (2) incomplete scan design rule checking and repair in which the asynchronous set/reset signals of some scan cells are not disabled during the shift operation or the gated/generated clocks for some scan cells are not properly enabled or disabled; or (3) incorrect scan synthesis in which positive-edge scan cells are placed before negative-edge scan cells.

Scan capture problems typically occur because of mismatches between the zero-delay model used in test generation and fault simulation tools and the full-timing behavior of the real device. In these cases, care must be taken during the scan design and test application process to (1) provide enough clock delay between the supplied clocks such that the clock capture order becomes deterministic, and (2) prevent simultaneous clock and data switching events from occurring. Failure to take clock events into proper consideration can easily result in a breakdown of the zero-delay (cycle-based) simulator used in the test generation and fault simulation process. More detailed information regarding scan verification of the shift and capture operations is described in the following.

7.2.2.4.1 Verifying the scan shift operation

Verifying the scan shift operation involves performing **flush tests** with a full-timing logic simulator during the shift operation. A flush test is a shift test in which a selected flush pattern is shifted all the way through the scan chains to verify that the same flush pattern arrives at the end of the scan chains at the correct clock cycle. For example, a scan chain containing 1000 scan cells requires 1000 shift cycles to be applied to the scan chain for the selected flush pattern to begin arriving at the scan output. If the data arrive early by a number of shift cycles, this may indicate that a similar number of hold time problems exist in the circuit.

To detect clock skew problems between adjacent scan cells, the selected flush pattern is typically a pattern that is capable of providing both 0-to-1 and

1-to-0 transitions to each scan cell. To ensure that a 0-to-0 or 1-to-1 transition of a scan cell does not corrupt the data, the selected flush pattern is further extended to provide these transitions. A typical flush pattern used for testing the shift operation is "01100," which includes all four possible transitions. Different flush patterns can also be used for debugging different problems, such as the all-zero and all-one flush patterns used for debugging stuck-at faults in the scan chain.

Because observing the arrival of the data on the scan chain output cannot pinpoint the exact location of any shift error in a faulty scan chain, flush testbenches are typically created to observe the values at all internal scan cells to identify the locations at which the shift errors exist. By use of this technique, the faulty scan chain can be easily and quickly diagnosed and fixed during the scan shift verification process; for example:

- Scan hold time problems that exist between scan cells belonging to different clock domains indicate that a lock-up latch may be missing. Lock-up latches should be inserted between these adjacent scan cells.
- Scan hold time and setup time problems that exist between scan cells belonging to the same clock domain indicate that the CTS process was not performed correctly. In this case, either the CTS has to be redone or additional buffers need to be inserted between the failing scan cells to slow down the path.
- Scan hold time problems caused by positive-edge scan cells followed by negative-edge scan cells indicate that the scan chain order was not performed correctly. Lock-up flip-flops rather than lock-up latches can be inserted between these adjacent scan cells or the scan chains may have to be reordered by placing all negative-edge scan cells before all positive-edge scan cells.

An additional approach to scan shift verification that has become more popular in recent years involves performing ***static timing analysis*** (STA) on the shift path in shift mode. In this case, the STA tool can immediately identify the locations of all adjacent scan cells that fail to meet timing. The same solutions mentioned earlier are then used to fix problems identified by the STA tool.

7.2.2.4.2 Verifying the scan capture operation

Verifying the scan capture operation involves simulating the scan design with a full-timing logic simulator during the capture operation. This is used to identify the location of any failing scan cells in which the captured response does not match the expected response predicted by the zero-delay logic simulator used in test generation or fault simulation. To reduce simulation time, a **broadside-load testbench** is often used, in which a test pattern is loaded directly into all scan cells in the scan chains and only the capture cycle is simulated. Because the **broadside-load test** does not involve any shift cycle in the test pattern, broadside-load testbenches often include at least one shift cycle in the capture verification testbench to ensure that each test pattern can at least shift once.

This requires loading the test pattern into the outputs of the previous scan cells rather than directly into the outputs of the current scan cells. In addition, verifying the scan capture operation often includes a *serial simulation*, in which a limited number of test patterns, typically three to five or as many as can be simulated within a reasonable time, are simulated. In this serial simulation, a test pattern is simulated exactly the same as how it would be applied on the tester by shifting in each pattern serially through the scan chain inputs. Next, a capture cycle is applied. The captured response is then shifted out serially to verify that the complete scan chain operation can be performed successfully.

As mentioned before, mismatches in the capture cycle indicate that the zero-delay simulation model used by the test generator and the fault simulator failed to capture all the details of the actual timing occurring in the device. Debugging these types of failures is tedious and may involve observing all signals of the mismatching scan cells and signal lines (also called nets) driving these scan cells. One brute-force method commonly used by designers for removing these mismatches is to mask off the locations by changing the expected response of the mismatching location into an unknown (X) value. A new approach that has become more popular is to use the static timing analysis tool for both scan shift and scan capture verification.

7.3 LOGIC BUILT-IN SELF-TEST (BIST) DESIGN

Compared with scan synthesis, considerable efforts are required to implement or synthesize a BIST design. Because logic BIST is mostly scan-based, the BIST design must first comply with all scan design rules. In addition, because BIST designs usually cannot tolerate unknown (X) values propagated to the output response analyzers, **BIST-specific design rules** are required to deal with unknown sources originating from analog blocks, memories, non-scan storage elements, asynchronous set/reset signals, tristate buses, false paths, and multiple-cycle paths, to name a few. The need to implement a logic BIST controller that automatically coordinates BIST pattern generation and response analysis for at-speed testing of delay faults further complicates the process. Last, because pseudo-random patterns (as opposed to deterministic patterns) are commonly used for BIST pattern generation, additional test points (including control points and observation points) may have to be added to improve the circuit's fault coverage.

7.3.1 BIST design rules

Because logic BIST requires many more stringent design restrictions than conventional scan, many *scan design rules* discussed in Section 7.2 that are optional for scan designs become mandatory for BIST designs. The major logic BIST design restriction relates to the propagation of unknown (X) values. Because any unknown (X) value that propagates directly or indirectly to the

output response analyzer (ORA) will corrupt the *signature* and cause the BIST signature to become useless, no unknown (*X*) values can be tolerated. This is different from scan designs in which unknown (*X*) values present in a scan design only result in fault coverage degradation. Therefore, when designing a logic BIST system, it is essential that the *circuit under test* (CUT) meet all scan design rules and BIST-specific design rules, called **BIST design rules**. The process of taking a scan-based design and making it meet all additional BIST-specific design rules turns the design into a **BIST-ready core**.

7.3.1.1 *Unknown source blocking*

There are many unknown (*X*) sources in a CUT or BIST-ready core. Any unknown (*X*) source in the BIST-ready core, which is capable of propagating its unknown (*X*) value to the ORA directly or indirectly, must be blocked and fixed with a DFT repair approach often called **X-bounding** or **X-blocking**. Figure 7.12 shows a few of the more typically used X-bounding methods for blocking an unknown (*X*) source: The **0-control point** forces an *X* source to 0; the **1-control point** controls the *X* source to 1; the **bypass logic** allows the output of the *X* source to receive both 0 and 1 from a *primary input* (PI) or an internal node; the **control-only scan point** drives both 0 and 1 through a storage element, such as D flip-flop; and finally, the **scan point** can capture the *X*-source value and drive both 0 and 1 through a scan cell, such as scan D flip-flop or ***level-sensitive scan design*** (LSSD) ***shift register latch*** (SRL) [Eichelberger 1977].

Depending on the nature of each unknown (*X*) source, several *X*-bounding methods can be appropriate for use. The most common problems inherent in these approaches include: (1) that they might increase the area of the design, and (2) that they might impact timing.

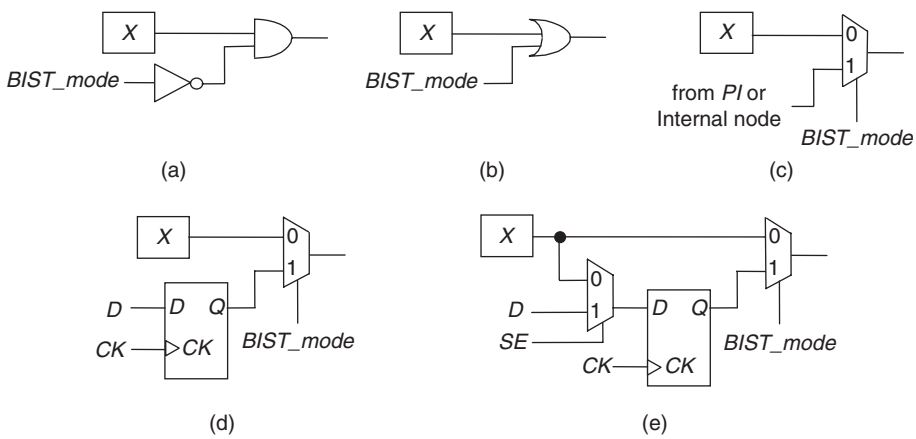


FIGURE 7.12

Typical X-bounding methods for blocking an unknown (*X*) source: (a) 0-control point. (b) 1-control point. (c) Bypass logic. (d) Control-only scan point. (e) Scan point.

7.3.1.1.1 Analog blocks

Examples of analog blocks are ***analog-to-digital* converters** (ADCs). Any analog block output that can exhibit unknown (*X*) behavior during a test has to be forced to a known value. This can be accomplished by adding a 0-control point, a 1-control point, bypass logic, or a control-only scan point. We recommend the latter two approaches, because they yield higher fault coverage than the former two approaches.

7.3.1.1.2 Memories and non-scan storage elements

Examples of memories are *dynamic random-access memories* (DRAMs), *static random-access memories* (SRAMs), or flash memories. Examples of non-scan storage elements are D flip-flops or D latches. Bypass logic is typically used to block each unknown (*X*) value originating from a memory or non-scan storage element. Another approach is to use an initialization sequence to set a memory or non-scan storage element to a known state. This is typically done to avoid adding delay to critical (functional) paths. Care must be taken to ensure that the stored state is not corrupted throughout the BIST operation.

7.3.1.1.3 Combinational feedback loops

All combinational feedback loops must be avoided. If they are unavoidable, then each loop must be broken with a 0-control point, a 1-control point, or a scan point. We recommend adding scan points because they yield higher fault coverage than the other approaches.

7.3.1.1.4 Asynchronous set/reset signals

As indicated in the preceding section, asynchronous set or reset can destroy the data during the shift operation if a pattern causes the set/reset signal to become active. The asynchronous set or reset can be disabled with an external set/reset disable (*RE*) pin given in Figure 7.7. This set/reset disable pin must be set to 1 during the shift operation. This may become cumbersome for BIST applications in which there is a need to use the pin for other purposes. Thus, we recommend using the existing scan enable (*SE*) signal to protect each shift operation and adding a set/reset clock point (*SRCK*) on each set/reset signal to test the set/reset circuitry as illustrated in Figure 7.13.

In addition, we recommend testing all data and set/reset faults with two separate BIST sessions as shown in Figure 7.14. The timing diagram in this figure is used for testing a circuit having one system clock (*CK*) and one added set/reset clock. To test data faults in the functional logic, a clock pulse C_1 is triggered from *CK* while *SRCK* is held inactive in one capture window. Similarly, to test set/reset faults in the set/reset circuitry, C_2 is enabled while *CK* is held inactive in another capture window. By use of this approach, we can avoid races and hazards and prevent data in scan cells from being destroyed by the set/reset signals.

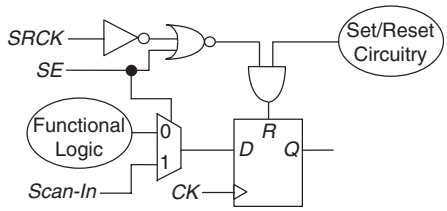


FIGURE 7.13

Set/reset clock point for testing a set/reset-type scan cell.

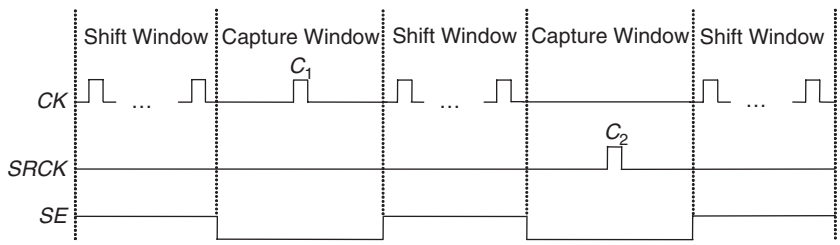


FIGURE 7.14

Example timing control diagram for testing data and set/reset faults.

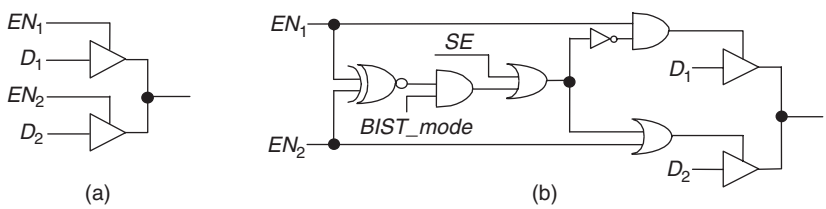


FIGURE 7.15

A one-hot decoder for testing a tristate bus with two drivers: (a) A tristate bus. (b) A one-hot decoder.

7.3.1.1.5 Tristate buses

Bus contention occurs when two drivers force different values on the same bus that can damage the chip; hence, it is important to prevent bus conflicts during the normal operation and the shift operation [Cheung 1997]. For BIST applications, because pseudo-random patterns are commonly used, it is also crucial to prevent bus contention from happening during the capture operation [Al-Yamani 2002]. To avoid potential bus contention, it is best to resynthesize each bus with multiplexers. If this is impractical, make sure only one tristate driver is enabled at any given time. The **one-hot decoder** shown in Figure 7.15 is an example of a circuit that can ensure only one driver is selected during each shift or capture operation.

7.3.1.1.6 False paths

False paths are not normal functional paths. They do no make any harm to the chip during the normal operation; however, for delay fault testing, a pseudo-random pattern might adversely attempt to test a selected false path. Because false paths are not exercised during the normal circuit operation, they typically do not meet timing specifications, which can result in a mismatch during logic BIST delay fault testing. To avoid this potential problem, we recommend adding a 0-control point or 1-control point to each false path.

7.3.1.1.7 Critical paths

Critical paths are timing-sensitive functional paths. Because the timing of such a path is critical, no extra gates are allowed to be added to the path to prevent increasing the delay of the critical path. To remove an unknown (X) value from a critical path, we recommend adding an extra input pin to a selected combinational gate, such as an inverter, NAND gate, or NOR gate, on the critical path to minimize the added delay. The combinational gate is then converted to an embedded 0-control point or embedded 1-control point as shown in Figure 7.16, where an inverter is selected for adding the extra input.

7.3.1.1.8 Multiple-cycle paths

Multiple-cycle paths are normal functional paths, but data are expected to arrive after two or more cycles. Similar to false paths, they can cause mismatches if exercised during delay fault testing, because they are intended to be tested in one cycle. To avoid this potential problem, we recommend adding a 0-control point or 1-control point to each multiple-cycle path or holding certain scan cell output states to avoid those multiple-cycle paths.

7.3.1.1.9 Floating ports

Neither *primary inputs* (PIs) nor *primary outputs* (POs) can be floating. These ports must have a proper connection to power (V_{DD}) or ground (V_{SS}). Also, floating inputs to any internal modules must be avoided. This has a potential chance to propagate unknown (X) values to the ORA.

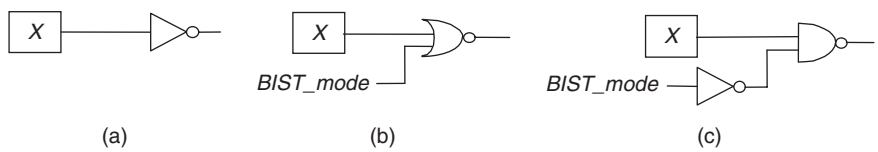


FIGURE 7.16

Embedded control points for testing a critical path having an inverter: (a) An inverter. (b) Embedded 0-control point. (c) Embedded 1-control point.

7.3.1.1.10 Bidirectional I/O ports

Bidirectional I/O ports are commonly used in a design. For BIST operations, the direction of each bidirectional I/O port should be forced to either input or output mode. Figure 7.17 shows an example of forcing a bidirectional I/O port to output mode.

7.3.1.2 *Re-timing*

Because the TPG and the ORA are typically placed far from the CUT, races and hazards caused by clock skews may occur between the TPG and the (scan chain) inputs of the CUT and between the (scan chain) outputs of the CUT and the ORA. To avoid these potential problems and ease physical implementation, we recommend adding **re-timing logic** between the TPG and the CUT and between the CUT and the ORA. The re-timing logic should consist of at least one negative-edge **pipelining register** (D flip-flop) and one positive-edge pipelining register (D flip-flop). Figure 7.18 shows an example re-timing logic among the TPG, CUT, and ORA that uses two pipelining registers on each end. Note that the three clocks (CK_1, CK_2, and CK_3) could belong to one clock tree.

7.3.2 BIST design example

In this subsection, we show an example of designing a logic BIST system for testing a scan-based design (core) composed of two clock domains with s38417 and s38584. The two clock domains are taken from the ISCAS-1989 benchmark circuits [Brglez 1989], and their statistics are shown in Table 7.2. The design we consider is described at the *register-transfer level* (RTL). We

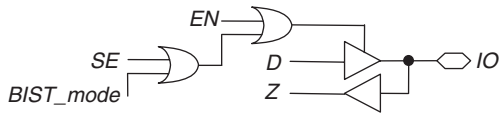


FIGURE 7.17

Forcing a bidirectional port to output mode.

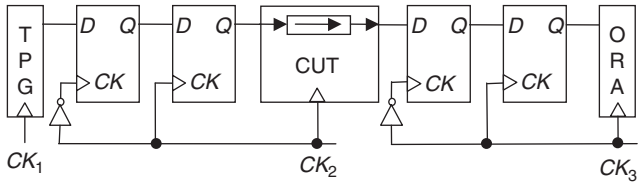


FIGURE 7.18

Re-timing logic among the TPG, CUT, and ORA.

Table 7.2 Design Statistics

Clock Domain	No. of PIs	No. of POs	No. of Flip-Flops	No. of Gates
CD_1 (s38417)	28	106	1,636	22,179
CD_2 (s38584)	12	278	1,452	19,253

show all the necessary steps to arrive at the logic BIST system design, verify its correctness, and improve its fault coverage.

7.3.2.1 *BIST rule checking and violation repair*

The first step is to perform logic BIST design rule checking on the RTL design. All DFT rule violations of the *scan design rules* and *BIST-specific design rules* provided in preceding sections must be repaired. Once all DFT rule violations are repaired, the design should meet all scan and logic BIST design rules. In addition, we should be aware of the following design parameters:

- The number of test clocks present in the design, each used for controlling one clock domain.
- The number of set/reset clocks present in the design to be used for breaking all asynchronous set/reset loops.

In the preceding example, the design contains two test clocks and does not require any additional set/reset clock. The new RTL design (core) after BIST rule repair is performed is referred to as an *RTL BIST-ready core*.

7.3.2.2 *Logic BIST system design*

The second step is to design the logic BIST system at the RTL. The decisions that need to be made at this stage include:

- The type of logic BIST architecture to adopt.
- The number of PRPG-MISR (or PEPG-MISR) pairs to use
- The length of each PRPG-MISR (or PEPG-MISR) pair.
- The faults to be tested and BIST timing control diagrams to be used for testing these faults.
- The types of optional logic to be added to ease physical implementation and facilitate debug and diagnosis, as well as improve the circuit's fault coverage.

7.3.2.2.1 Logic BIST architecture

We choose to implement **STUMPS**-based logic BIST architecture, because it is easy to integrate with scan/ATPG and is the architecture widely used in the industry. We recommend the use of one PRPG-MISR pair for each clock domain, whenever possible, because the resulting BIST architecture is easier to debug.

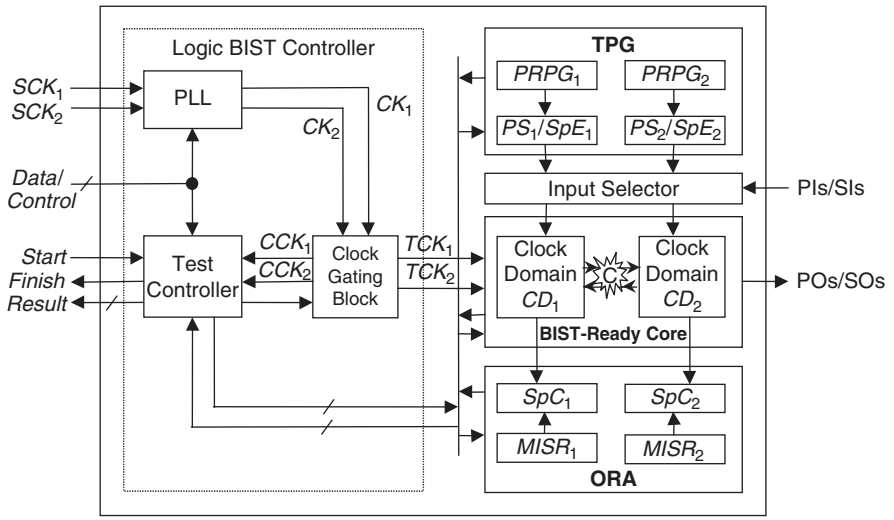


FIGURE 7.19

A logic BIST system for testing a design with two cores.

In addition, the use of one PRPG-MISR pair for each clock domain can eliminate the need for additional design efforts for managing clock skews between interacting clock domains, even when they operate at the same frequency. If it is required to use a single PRPG-MISR pair to test multiple clock domains, these clock domains should be placed within physical proximity to simplify physical implementation. An example of logic BIST system based on the STUMPS architecture for testing the design given in Table 7.2 is shown in Figure 7.19.

The BIST architecture used for testing the BIST-ready core consists of a TPG for generating test stimuli, an input selector for providing pseudo-random or ATPG patterns to the core-under-test, an ORA for compacting the test responses, and a logic BIST controller for coordinating the overall BIST operation. The logic BIST controller consists of a test controller and a clock-gating block. The test controller initiates the BIST operation on receiving a *Start* signal, issues a *Finish* signal once the BIST operation is complete, and reports the pass/fail status of the test through the *Result* bus. The clock-gating block accepts internal PLL clocks (CK_1 and CK_2) derived from external functional clocks (SCK_1 and SCK_2), and generates the required test clocks (TCK_1 and TCK_2) and controller clocks (CCK_1 and CCK_2) for controlling the BIST-ready core and test controller, respectively. During normal functional operation, both CK_1 and CK_2 can run faster or slower than SCK_1 and SCK_2, respectively.

7.3.2.2.2 TPG and ORA

Next, we need to determine the length of each PRPG-MISR pair. The use of a separate PRPG-MISR pair for each clock domain allows us to reduce the length

of each PRPG and MISR. In the example shown in Figure 7.19, the linear phase shifters, PS_1 and PS_2, and space expanders, SpE_1 and SpE_2, can be used to further reduce the length of the PRPGs, whereas the space compactors, SpC_1 and SpC_2, can be used to further reduce the length of the MISRs. Each space expander or space compactor typically consists of an XOR-gate tree.

Now, suppose we decide to (1) synthesize the two clock domains, CD_1 and CD_2, each with 20 balanced scan chains; (2) run 100,000 pseudo-random patterns to obtain very high BIST fault coverage by adding additional test points; and (3) perform top-up ATPG after BIST to further increase the circuit's fault coverage. Because CD_1 has 28 PIs, a logical conclusion would be to expect the length of the $PRPG_1$ to be 48 for the use of a 48-stage PRPG to drive 28 PIs and 20 scan chains. Because we plan to perform top-up ATPG, which requires sharing 20 out of the 28 PIs with ***scan inputs*** (SIs), and another 20 POs with ***scan outputs*** (SOs), another possible length for the $PRPG_1$ would be 28. What we need to determine is whether a 28-stage PRPG, constructed from a maximum-length LFSR or ***cellular automata*** (CA), is adequate for generating the required 100,000 pseudo-random patterns.

For a CD_1 with 20 balanced scan chains, 82 shift clock pulses are required (1636 flip-flops/20 scan chains) to scan in a single pseudo-random pattern. This means that a total of 8.2 million shift clock pulses are required to scan in all 100,000 patterns. This number is much smaller than the 256 million ($2^{28}-1$) patterns generated with a 28-stage maximum-length LFSR or CA for the $PRPG_1$. From Table 3.5 given in Chapter 3, we chose a 28-stage maximum-length LFSR with characteristic polynomial, $f(x) = 1 + x^3 + x^{28}$.

A similar analysis applies for CD_2. The main difference is that CD_2 has 12 PIs. Suppose we pick 10 out of the 12 PIs to share with 10 SIs for top-up ATPG. We will need to use a 10-to-20 space expander (SpE_2) for driving the 20 scan chains and a 20-to-10 space compactor (SpC_2) for driving the 10 SOs. Because testing this clock domain requires a total of 7.3 million (1452/20 $\times$ 00,000) shift clock pulses, we need to use at least a 23-stage maximum-length LFSR or CA as $PRPG_2$ to drive the 12 PIs. From Table 3.5 given in Chapter 3, we chose a 25-stage maximum-length LFSR with characteristic polynomial, $f(x) = 1 + x^3 + x^{25}$.

As indicated in Section 3.4.2.3, each MISR can cause an *aliasing* problem, but the problem is of less concern when the MISR length is greater than 20. Because CD_1 and CD_2 both have 106 and 278 POs, we choose a 106-to-27 space compactor (SpC_1) and a 278-to-35 space compactor (SpC_2), respectively. Thus, we will use a 47-stage MISR and a 45-stage MISR to compact the test responses from both CD_1 and CD_2, respectively, where 47 = 27 (shared POs) + 20 (SOs) and 45 = 35 (shared POs) + 10 (SOs). From Table 3.5 given in Chapter 3, we choose to implement the 47-stage MISR with $f(x) = 1 + x^5 + x^{47}$, and the 45-stage MISR with $f(x) = 1 + x + x^3 + x^4 + x^{45}$. Table 7.3 shows the decisions made for each PRPG-MISR pair so far.

Table 7.3 PRPG-MISR Choices

Clock Domain	No. of Scan Chains	No. of Shared SIs or SOs	Max. Scan Chain Length	PRPG Length	MISR Length
CD_1 (s38417)	20	20	82	28	47
CD_2 (s38584)	20	10	73	25	45

7.3.2.2.3 Test controller

The test controller plays a central role in coordinating the overall BIST operation. In general, a *finite-state machine* written at the RTL is used to implement the test controller for interfacing with all external signals, such as *Start*, *Finish*, and *Result*, and generating the required timing control signals for controlling each PRPG-MISR pair and the BIST-ready core. Comparison logic is included in the test controller to compare the *final signature* with an embedded *golden signature*.

Often, these interface signals are controlled through an IEEE 1149.1 boundary-scan-standard-based ***test access port* (TAP) *controller*** [IEEE 1149.1-2001]. In this case, all signals can be assessed through the TAP: TDI (*Test Data In*), TDO (*Test Data Out*), TCK (*Test Clock*), and TMS (*Test Mode Select*). Optionally, an IEEE 1500 standard-based **wrapper** may be also used to isolate each selected clock domain [IEEE 1500-2005].

To test structural faults in the BIST-ready core, we chose the *staggered single-capture* approach rather than the *one-hot single-capture* approach. The slow-speed timing control diagram is shown in Figure 7.20, where test clocks TCK_1 and TCK_2 are staggered and generated by the clock-gating block shown in Figure 7.19.

To test delay faults in the BIST-ready core, we chose the *staggered double-capture* approach if CD_1 and CD_2 are asynchronous or the *aligned double-capture* approach if they are synchronous. This is because either approach

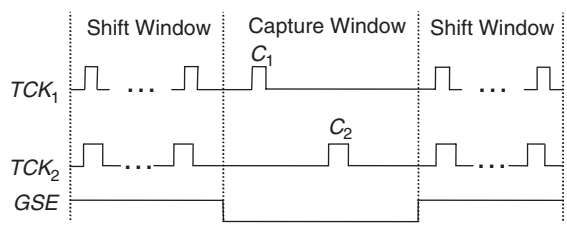


FIGURE 7.20

Slow-speed timing control using staggered single-capture.

allows us to operate a *global scan enable* (*GSE*) signal at slowspeed for driving all clock domains simultaneously in both BIST and scan ATPG modes. The at-speed timing control diagrams with the *staggered double-capture* and *launch aligned double-capture* schemes are shown in Figures 7.21 and 7.22, respectively.

7.3.2.2.4 Clock gating block

To generate an ordered sequence of *single-capture* or *double-capture* clocks, *clock suppression* [Rajski 2003], *daisy-chain clock-triggering*, or *token-ring clock-enabling* [Wang 2005a] can be used. The clock suppression scheme typically requires the use of a reference clock operating at the highest frequency. Daisy-chain clock-triggering means that a completion of one event automatically triggers the next event as the arrows shown in Figure 7.23. The only difference

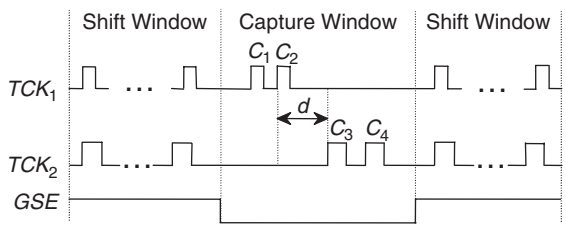


FIGURE 7.21

At-speed timing control using staggered double-capture.

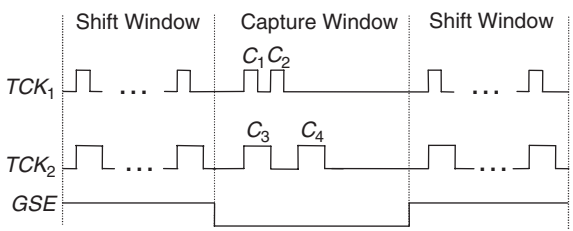


FIGURE 7.22

At-speed timing control using launch aligned double-capture.

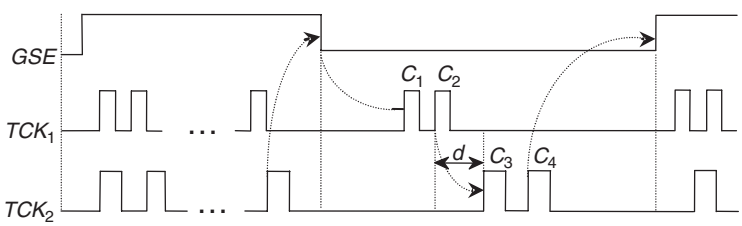


FIGURE 7.23

Daisy-chain clock-triggering.

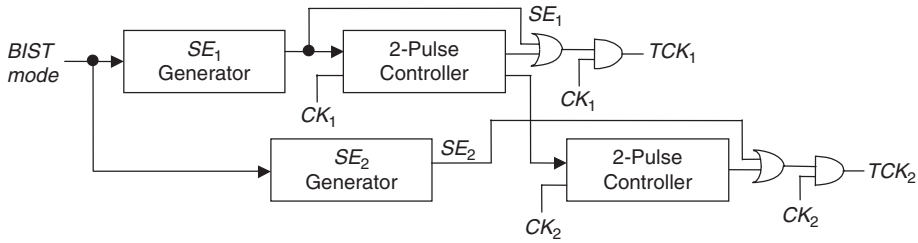


FIGURE 7.24

A daisy-chain clock-triggering circuit for generating the waveform given in Figure 7.23.

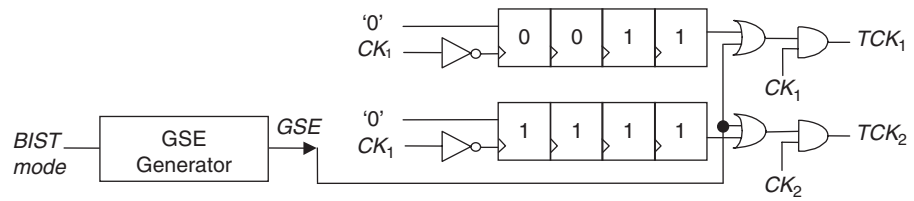


FIGURE 7.25

A clock suppression circuit for generating the waveform given in Figure 7.22.

between daisy-chain clock-triggering and token-ring clock-enabling is that the former uses a clock edge to trigger the next event, whereas the latter uses a signal level to enable the next event.

Figure 7.24 shows a daisy-chain clock-triggering circuit for generating the *staggered double-capture* waveform given in Figure 7.23. When the BIST mode is activated, the SE_1/SE_2 generators and 2-pulse controllers will generate the required scan enable and double-capture clock pulses, per the arrows shown in Figure 7.23. Each SE_1/SE_2 can be treated as a *GSE* signal for CD_1/CD_2.

Figure 7.25 shows a clock suppression circuit for generating the *launch aligned double-capture* waveform given in Figure 7.22. This circuit uses a reference clock (CK_1) to program the capture window. The contents of the 8-bit shift register are preset to {0011,1111} during each shift window. Because of its programmability, the approach can also be used to generate timing waveforms for testing asynchronous designs. One major requirement is that we guarantee that the delay measured by the number of reference clock pulses be longer than delay d between C_2 and C_3, as shown in Figure 7.21.

7.3.2.2.5 Re-timing logic

The main difference between ATE-based scan testing and logic BIST is that the latter requires that more complex BIST circuitry be implemented on the

functional circuitry. Successfully completing the physical implementation of the functional circuitry of a high-speed and high-performance design is a challenge in itself. If the BIST circuitry adds a large number of timing critical signals and requires strict clock-skew management, the physical implementation of logic BIST can become extremely difficult. Therefore, we recommend adding two pipelining registers (see Figure 7.18) between each PRPG and the BIST-ready core and two additional pipelining registers between the BIST-ready core and each MISR. In this case, the maximum scan chain length for each clock domain, CD_1 or CD_2, is effectively increased by 2, not 4.

7.3.2.2.6 Fault coverage enhancing logic and diagnostic logic

The drawback of using pseudo-random patterns is that the circuit may not meet the target fault coverage goal. To improve the circuit's fault coverage, we recommend adding extra test points and additional logic for top-up ATPG support at the RTL. A general rule of thumb is to add one extra test point for every 1000 gates. For top-up ATPG support, the inserted logic includes an input selector for selecting test patterns either from the PRPGs or PIs/SIs, as shown in Figure 7.19, as well as circuitry for reconfiguring the scan chains to perform top-up ATPG in (1) ATPG mode or (2) ATPG compression mode, which is discussed in more detail in Chapter 3.

We also recommend including **diagnostic logic** in the RTL BIST code to facilitate debug and diagnosis [Wang 2006b]. One simple approach is to connect all PRPG-MISR pairs (and all scan chains) as a serial scan chain and make them externally accessible. (Refer to Chapter 7 [Wang 2006a] for more advanced BIST diagnosis techniques.) Table 7.4 summarizes all possible test modes of the logic BIST system along with the effective scan chain counts for each test mode.

7.3.2.3 ***RTL BIST synthesis***

Once all decisions regarding the logic BIST architecture are made, it is time to create the RTL logic BIST code. At this stage, it is possible to either design the

Table 7.4 Example Test Modes Supported by the Logic BIST System

Test Mode	CD1 Effective Chain Count	CD2 Effective Chain Count
Normal	0	0
BIST	20	20
ATPG	20	10
ATPG compression	20	20
Serial debug and diagnosis	1	1

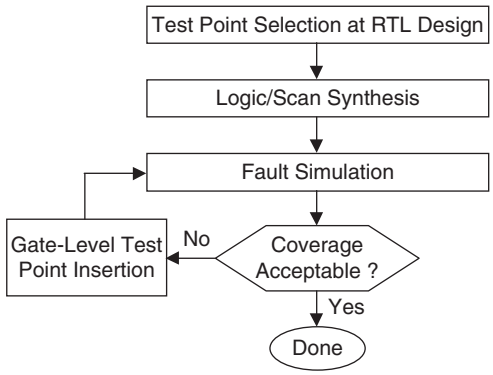


FIGURE 7.26

Fault simulation and test point insertion flow.

logic BIST system by hand or generate the RTL code automatically with an RTL logic BIST tool (commercially available). In either case, the number of scan chains for each clock domain should be specified along with the names of their associated scan inputs and scan outputs without inserting the actual scan chains into the circuit. The scan synthesis task can be handled as part of the general synthesis task, implemented with any commercially available synthesis tool for converting the RTL BIST-ready core and the logic BIST system into a gate-level netlist.

7.3.2.4 *Design verification and fault coverage enhancement*

Finally, the synthesized netlist needs to be verified with functional or timing verification to ensure that the logic BIST system functions as intended. If any pattern mismatch occurs, the problem must be identified and resolved. Next, fault simulation must be performed on the pseudo-random patterns generated by the TPG to determine the circuit's fault coverage. If the circuit does not reach the target fault coverage goal, additional test points should be inserted or top-up ATPG should be used. The extra test points that were added in advance at the RTL design should allow you to achieve the target fault coverage goal; otherwise, the test point insertion and fault simulation process may have to be repeated until the final fault coverage goal is reached. Once this process is complete, the *golden signature* can be either recorded to be compared externally or hard-coded into the comparison logic. The fault simulation and test point insertion flow is illustrated in Figure 7.26.

7.4 RTL DESIGN FOR TESTABILITY

During the 1990s, the testability of a circuit was primarily assessed and improved at the gate level. The reason was because the circuits were not too

large that the logic/scan synthesis process took an unreasonable amount of time. As device size grows toward tens to hundreds of millions of transistors, tight timing, potential yield loss, and low power issues begin to pose serious challenges. When combined with increased core reusability and time-to-market pressure, it is becoming imperative that most, if not all, testability issues be addressed at the RTL. This allows the logic/scan synthesis tool and the physical synthesis tool, which takes physical layout information into consideration, to optimize area, power, and timing after DFT repairs are made. Fixing DFT problems at the RTL also allows designers to create testable RTL cores that can be reused without having to repeat the DFT checking and repair process for a number of times.

Figure 7.27 shows a design flow for performing testability repair at the gate level. It is clear that performing testability repair at the gate level introduces a loop in the design flow that requires repeating the time-consuming logic synthesis process every time testability repair is made. This makes it logical to attempt to perform testability checking and repair at the RTL instead, so testability violations can be detected and fixed at the RTL, as shown in Figure 7.28, without having to repeat the logic synthesis process.

An additional benefit of performing testability repair at the RTL is that it allows scan to be more easily integrated with other advanced DFT features implemented at the RTL, such as memory BIST, logic BIST, test compression, boundary scan, and *analog and mixed-signal* (AMS) BIST. This makes it possible to perform all testability integration at the RTL as opposed to the current practices of integrating the advanced DFT features at the RTL and later integrating them with scan at the gate level. In the following, we describe the RTL DFT problems by focusing mainly on scan design.

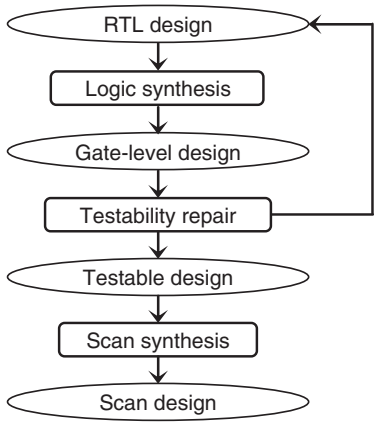


FIGURE 7.27

Gate-level testability repair design flow.

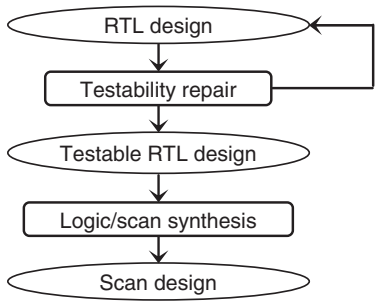


FIGURE 7.28

RTL testability repair design flow.

Some modern synthesis tools now incorporate testability repair and scan synthesis as part of the logic synthesis process, such that a testable design free of scan rule violations is generated automatically. In this case, if the DFT fixes made are acceptable and do not have to be incorporated into the RTL, the flow can proceed directly to test generation and scan verification.

7.4.1 RTL scan design rule checking and repair

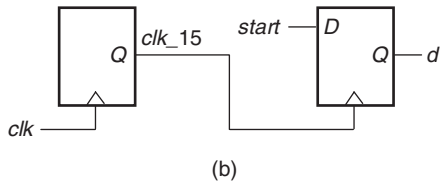
To perform scan design rule checking and repair at the RTL, a **fast synthesis** step of the RTL is usually performed first. In fast synthesis, combinational RTL code is mapped onto combinational primitives and high-level models, such as adders and multipliers. This allows us to identify all possible scan design rule violations and infer all storage elements in the RTL design.

Static solutions for identifying testability problems at the RTL without having to perform any test vector simulation or dynamic solutions that simulate the structure of the design through the RTL have been developed. These solutions make it possible to identify almost all testability problems at the RTL. Although a few testability problems remain that can be identified only at the gate level, this approach does reduce the number of iterations involving logic synthesis, as shown in Figure 7.28. In addition, it has become common to add scan design rules as part of RTL "*lint*" tools that check for good coding and reusability styles, as well as user-defined coding style rules [Keating 1999]. To further optimize testability results, *clock grouping* can also be performed at the RTL as part of scan design rule checking [Wang 2005b].

Automatic methods for repairing RTL testability problems have also been developed [Wang 2005b]. An example is shown in Figure 7.29. The RTL code shown in Figure 7.29a, which is written in the Verilog ***Hardware Description Language*** (HDL) [IEEE 1463-2001], represents a generated clock. In this example, a flip-flop *clk*_15 can be inferred, whose value is driven to 1 when a counter value *q* is equal to "1111." The output of this flip-flop is then used to trigger the second "always" statement, where an additional flip-flop can be

```
always @(posedge clk)
   if (q == 4'b1111)
      clk_15 <= 1;
   else
      begin
         clk_15 <= 0;
         q <= q + 1;
      end
always @(posedge clk_15)
   d <= start;
```

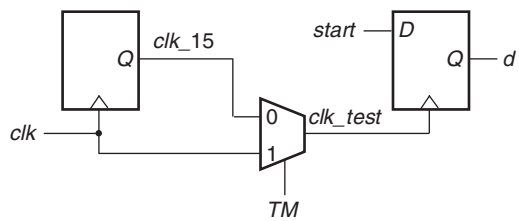
(a)



(b)

```
always @(posedge clk)
   if (q == 4'b1111)
      clk_15 <= 1;
   else
      begin
         clk_15 <= 0;
         q <= q + 1;
      end
assign clk_test = (TM) ? clk : clk_15;
always @(posedge clk_test)
   d <= start;
```

(c)



(d)

FIGURE 7.29

Automatic repair of a generated clock violation at the RTL: (a) Generated clock (RTL code). (b) Generated clock (schematic). (c) Generated clock repair (RTL code). (d) Generated clock repair (schematic).

inferred. Figure 7.29b shows a schematic of the flip-flop generating the *clk*_15 signal, as well as the flip-flop driven by the generated clock, which is likely to be the structure synthesized out of the RTL with a logic synthesis tool. This scan design rule violation can be fixed with the test mode signal *TM* by modifying the RTL code as shown in Figure 7.29c. The schematic for the modified RTL code is shown in Figure 7.29d.

7.4.2 **RTL scan synthesis**

When storage elements have been identified during RTL scan design rule checking, either **RTL scan synthesis** or **pseudo RTL scan synthesis** can be performed. In RTL scan synthesis, the scan synthesis step as described in Section 7.2.2.2 is performed. The only difference is that the scan equivalent of each storage element does not refer to a library cell but to an RTL structure that is equivalent to the original storage element in normal mode. In this case, the scan chains are inserted into the RTL design. In pseudo RTL scan synthesis, the scan synthesis step is not performed; only pseudo primary inputs and pseudo

primary outputs are specified and stitched to primary inputs and primary outputs, respectively. This approach is becoming more appealing to designers now, because it can cope with many advanced DFT structures, such as logic BIST and test compression, where scan chains are driven internally by additional test structures synthesized at the RTL. Once all advanced DFT structures are inserted at the RTL, a one-pass or single-pass synthesis step is performed with the RTL design flow as shown in Figure 7.28.

Several additional steps are actually performed to identify the storage elements in the RTL design. First, all clocks are identified, either explicitly by tracing from specified clock signal names or implicitly by analyzing the sensitivity list of all "always" blocks. When the clocks have been identified, all registers, each consisting of one or more storage elements in the RTL design, are inferred by analyzing all "assign" statements to determine which assignments can be mapped onto a register while keeping track of the clock domain to which each register belongs. In addition, the clock polarity of each register is determined.

After all registers have been identified and each converted into its scan equivalent at the RTL, the next step is to stitch these individual scan cells into one or more scan chains. One approach is to allocate scan cells to different scan chains on the basis of the driving clocks and to stitch all scan cells within a scan chain in a random fashion [Aktouf 2000]. Although this approach is simple and straightforward, it can introduce wiring congestion as well as high interconnect area overhead. To solve these issues, it is better to take full advantage of the rich functional information available at the RTL [Roy 2000; Huang 2001]. Because storage elements are identified as registers as opposed to a large number of unrelated individual storage elements, it is beneficial to connect the scan cells (which are scan equivalent of these storage elements) belonging to the same register sequentially in a scan chain. This has been found to be able to dramatically reduce wiring congestion and interconnect area overhead.

7.4.3 **RTL scan extraction and scan verification**

To verify the scan-inserted RTL design (also called *RTL scan design*), both scan extraction and scan verification must be performed. Scan extraction relies on performing fast synthesis on the RTL scan design. This generates a software model where scan extraction can be performed by tracing the scan connections of each scan chain in a similar manner as scan extraction from a *gate-level scan design*. Scan verification relies on a flush testbench that is used to simulate flush tests on the RTL scan design. Because the inputs and outputs of the RTL scan design should match the inputs and outputs of its gate-level scan design, the same flush testbench can be used to verify the scan operation for both RTL and gate-level designs. It is also possible to apply broadside-load tests for verifying the scan capture operation at the RTL. In this case, either random test patterns or deterministic test patterns generated at the RTL can be used [Ghosh 2001; Ravi 2001; Zhang 2003].

7.5 CONCLUDING REMARKS

This chapter has discussed the design rules and test synthesis steps required to implement the basic *design-for-testability* (DFT) techniques presented in Chapter 3 into modern digital circuits. By modeling the circuit at the gate level or the *register-transfer level* (RTL), modern test synthesis programs can perform design rule checking and repair before scan synthesis (including synthesis of test compression logic) or logic *built-in self-test* (BIST) synthesis. Modern RTL (logic) synthesis programs can further incorporate test synthesis into the logic synthesis flow.

In this chapter, we have presented a comprehensive discussion of scan synthesis. This includes scan design rules and a typical scan design flow. We have also provided a comprehensive description of scan-based logic BIST synthesis. This includes BIST-specific design rules and a BIST design example. These BIST-specific design rules are mandatory for logic BIST in addition to following all scan design rules. The RTL DFT techniques that include RTL scan design rule checking and RTL scan synthesis were briefly touched on at the end of the chapter; these techniques were used to enable DFT integration at the RTL.

Implementing testability logic in a design could require many dedicated test pins. Modern test synthesis programs have incorporated support of a few IEEE-endorsed standards into the test synthesis flow to reduce additional pin count or to facilitate test, debug, and diagnosis. The most popular standard supported is the IEEE 1149.1 boundary-scan standard [IEEE 1149.1-2001], because it requires only four or five dedicated pins regardless of what testability logic is to be implemented. A few others that start to gain popularity include the IEEE 1149.6 boundary-scan standard for advanced digital networks [IEEE 1149.6-2003] and the IEEE 1500 embedded core-test standard [IEEE 1500-2005]. For more information on these emerging standards, refer to [Wang 2006a, 2007].

7.6 EXERCISES

7.1. (**Lock-Up Latch**) Suppose that a scan chain is configured as $SI \rightarrow SFF_1 \rightarrow SFF_2 \rightarrow SFF_3 \rightarrow SFF_4 \rightarrow SFF_5 \rightarrow SO$, where SFF_1 through SFF_5 are muxed-D scan cells, and *SI* and *SO* are the scan input pin and scan output pin, respectively. Suppose that this scan chain fails scan shift verification, in which the flush test sequence $<t_1\ t_2\ t_3\ t_4\ t_5> = <01010>$ is applied but the response sequence is $<r_1\ r_2\ r_3\ r_4\ r_5> = <01100>$. Identify the scan flip-flops that may have caused this failure, and show how to fix this problem by use of a lock-up latch.

7.2. (**Lock-Up Latch**) A scan chain may contain both positive-edge-triggered and negative-edge-triggered muxed-D scan cells. If, by accident, all positive-edge-triggered scan cells are placed before all

negative-edge-triggered muxed-D scan cells, show how to stitch them into one single scan chain. (*Hint*: Positive-edge–triggered muxed-D scan cells and negative-edge–triggered muxed-D scan cells should be placed into two separate sections.)

7.3. (**Lock-Up Latch**) Refer to Figure 7.11. The scheme works only when the clock skew between CK_1 and CK_2 is less than the width (duty cycle) of the clock pulse. If CK_2 is delayed more than the duty cycle of CK_1 (*i.e.*, CK_1 and CK_2 become nonoverlapping), show whether or not it is possible to stitch the two cross-clock-domain scan cells into one single scan chain with a lock-up latch. If not, can it be done with a lock-up flip-flop instead?

7.4. (**Scan Stitching**) Use examples to show why a scan chain may not be able to perform the shift operation properly if two neighboring scan cells in the scan chain are too close to or too far from each other. Also describe how to solve these problems.

7.5. (**Test Signal**) Describe the difference between the test mode signal *TM* and the scan enable signal *SE* used in scan testing.

7.6. (**Clock Grouping**) Show an algorithm to find the smallest number of clock groups in clocking grouping.

7.7. (**BIST Design Rules**) A scan design can contain many asynchronous set/reset signals that may require adding two or more set/reset clock points to break all ripple set/reset loops. A ripple set/reset loop is a combinational feedback loop. Assume that the design now contains two system clocks (CK_1 and CK_2) and two set/reset clocks ($SRCK_1$ and $SRCK_2$). Derive two BIST timing control diagrams, including a scan enable (*SE*) signal, to test all data faults and set/reset faults controlled by these four clocks. Explain which timing control diagram can detect more faults.

7.8. (**BIST Design Rules**) Design a one-hot decoder for testing a tristate bus with four independent tristate drivers in BIST mode.

7.9. (**BIST Design Rules**) Design an X-bounding circuit for improving the fault coverage of a bidirectional I/O port by forcing it to input mode during BIST operation.

7.10. (**Aligned Skewed-Load *versus* Aligned Double-Capture**) Assume there are four synchronous clock domains each controlled by a capture clock CK_1, CK_2, CK_3, or CK_4, and each operated at a frequency $F_1 = 2 \times F_2 = 4 \times F_3 = 8 \times F_4$. Derive BIST timing control diagrams with aligned skewed-load and aligned double-capture to test all intra-clock-domain and inter-clock-domain delay faults. Specify by arrows the delay faults that can be detected in the diagram.

7.11. (**Staggered Skewed-Load *versus* Staggered Double-Capture**) Assume there are four asynchronous clock domains each controlled by a capture clock CK_1, CK_2, CK_3, or CK_4, and each operated at a

frequency $F_1 > F_2 > F_3 > F_4$. Derive BIST timing control diagrams with staggered skewed-load and staggered double-capture to test all intra-clock-domain and inter-clock-domain delay faults. Specify by arrows the delay faults that can be detected in the diagram.

7.12. (Hybrid Double-Capture) Assume there are four mixed synchronous and asynchronous clock domains controlled by a capture clock, CK_1, CK_2, CK_3, and CK_4, operating at $F_1 = 100$ MHz, $F_2 = 50$ MHz, $F_3 = 60$ MHz, and $F_4 = 30$ MHz, respectively. Derive a BIST timing control diagram with a hybrid double-capture scheme composed of staggered double-capture and aligned double-capture to test all intra-clock-domain and inter-clock-domain delay faults. Specify by arrows the delay faults that can be detected in the diagram.

7.13. (RTL Testability Enhancement) Read the following Verilog HDL code and draw its schematic. Then determine whether there is any scan design rule violation. If there is any violation, then modify the RTL code to fix the problem, and draw the schematic of the modified RTL code.

```
reg [3:0] tri_en;
always @(posedge clk)
begin
  case (bus_sel)
  0: tri_en[0] = 1'b1;
  1: tri_en[1] = 1'b1;
  2: tri_en[2] = 1'b1;
  3: tri_en[3] = 1'b1;
  endcase
end
assign dbus = (tri_en[0])? d1 : 8'bz;
assign dbus = (tri_en[1])? d2 : 8'bz;
assign dbus = (tri_en[2])? d3 : 8'bz;
assign dbus = (tri_en[3])? d4 : 8'bz;
```

7.14. (A Design Practice) Use the scan design rule checking programs and user's manuals provided on the companion Web site to show whether you can detect any asynchronous set/reset signal violation and bus contention. Try to redesign a Verilog circuit to include such violations. Then, fix the violations by hand, and see whether the problems have disappeared.

7.15. (A Design Practice) Use the scan synthesis programs and user's manuals provided on the companion Web site to convert the two ISCAS-1989 benchmark circuits s27 and s38417 [Brglez 1989] into scan designs. Perform scan extraction and then run Verilog flush tests and broadside-load tests on the scan designs to verify whether the generated testbenches pass Verilog simulation.

7.16. (**A Design Practice**) Use the logic BIST programs and user's manuals provided on the companion Web site to design the logic BIST system with staggered double-capture for the circuit given in Section 7.3.2. Report the circuit's BIST fault coverage at every 10,000 increments up to 100,000 pseudo-random patterns.

7.17. (**A Design Practice**) Repeat Exercise 7.16, but instead implement the two pseudo-random pattern generators, PRPG1 and PRPG2, with a 28-stage CA and a 25-stage CA, respectively, with the construction rules given in Table 3.6. Explain why the CA-based logic BIST system can or cannot reach higher BIST fault coverage than the LFSR-based logic BIST system given in Exercise 7.16.

7.18. (**A Design Practice**) Use the ATPG programs and user's manuals provided on the companion Web site to report the circuit's ATPG fault coverage when the logic BIST system is reconfigured in ATPG mode. If the BIST fault coverage in Exercise 7.16 is lower than the ATPG fault coverage, insert as many test points as needed in the logic BIST system to reach the ATPG fault coverage; alternately, run top-up ATPG in both ATPG compression and ATPG modes and report the circuit's final fault coverage.

ACKNOWLEDGMENTS

We thank Khader S. Abdel-Hafez of Synopsys and formerly of SynTest Technologies for providing a portion of the materials in the Scan Design Flow section, Professor Wen-Ben Jone of the University of Cincinnati and Dr. Ravi Apte of SynTest Technologies for reviewing the chapter, and Teresa Chang of SynTest Technologies for drawing most of the figures.

REFERENCES

R7.0 Books

[Gizopoulos 2006] D. Gizopoulos, editor, *Advances in Electronic Testing: Challenges and Methodologies,* Morgan Kaufmann, San Francisco, 2006.

[Keating 1999] M. Keating and P. Bricaud, *Reuse Methodology Manual for System-on-a-Chip Designs*, Springer, Boston, 1999.

[Wang 2006a] L.-T. Wang, C.-W. Wu, and X. Wen, editors, *VLSI Test Principles and Architectures: Design for Testability,* Morgan Kaufmann, San Francisco, 2006.

[Wang 2007] L.-T. Wang, C. E. Stroud, and N. A. Touba, editors, *System-on-Chip Test Architectures: Nanometer Design for Testability*, Morgan Kaufmann, San Francisco, 2007.

R7.1 Introduction

[SIA 2005] SIA, *The International Technology Roadmap for Semiconductors: 2005 Edition*, Semiconductor Industry Association, San Jose, CA, http://public.itrs.net, 2005.

[SIA 2006] SIA, *The International Technology Roadmap for Semiconductors: 2006 Update*, Semiconductor Industry Association, San Jose, CA, http://public.itrs.net, 2006.

R7.2 Scan Design

[Barbagallo 1996] S. Barbagallo, M. Bodoni, D. Medina, F. Corno, P. Prinetto, and M. Sonza Reorda, Scan insertion criteria for low design impact, in *Proc. IEEE VLSI Test Symp.*, pp. 26–31, April 1996.

[Cheung 1997] B. Cheung and L.-T. Wang, The seven deadly sins of scan-based designs, *Integrated System Design*, www.eetimes.com/editorial/1997/test9708.html, August 1997.

[Duggirala 2002] S. Duggirala, R. Kapur, and T. W. Williams, System and Method for High-Level Test Planning for Layout, U.S. Patent No. 6,434,733, August 13, 2002.

[Duggirala 2004] S. Duggirala, R. Kapur, and T. W. Williams, System and Method for High-Level Test Planning for Layout, U.S. Patent No. 6,766,501, July 20, 2004.

R7.3 Logic Built-In Self-Test (BIST) Design

[Al-Yamani 2002] A. A. Al-Yamani, S. Mitra, and E. J. McCluskey, *Avoiding Illegal States in Pseudo-random Testing of Digital Circuits*, Center for Reliable Computing, Technical Report (CRC TR) No. 02-2, Stanford University, December 2002.

[Brglez 1989] F. Brglez, D. Bryan, and K. Kozminski, Combinational profiles of sequential benchmark circuits, in *Proc. IEEE Int. Symp. on Circuits and Systems*, pp. 1929–1934, August 1989.

[Cheung 1997] B. Cheung and L.-T. Wang The seven deadly sins of scan-based designs, *Integrated System Design*, www.eetimes.com/editorial/1997/test9708.html, August 1997.

[Eichelberger 1977] E. B. Eichelberger and T. W. Williams, A logic design structure for LSI testability, in *Proc. ACM/IEEE Design Automation Conf.*, pp. 462–468, June 1977.

[IEEE 1149.1-2001] IEEE Std. 1149.1-2001, *IEEE Standard Test Access Port and Boundary Scan Architecture*, IEEE Press, New York, 2001.

[IEEE 1500-2005] IEEE Std. 1500-2005, *IEEE Standard for Embedded Core Test*, IEEE Press, New York, 2005.

[Rajski 2003] J. Rajski, A. Hassan, R. Thompson, and N. Tamarapalli, Method and Apparatus for At-Speed Testing of Digital Circuits, U.S. Patent Application No. 20030097614, May 22, 2003.

[Wang 2005a] L.-T. Wang, X. Wen, P.-C. Hsu, S. Wu, and J. Guo, At-speed logic BIST architecture for multi-clock designs, in *Proc. IEEE Int. Conf. on Computer Design*, pp. 475–478, October 2005.

[Wang 2006b] L.-T. Wang, X. Wen, K. S. Abdel-Hafez, S.-H. Lin, H.-P. Wang, M.-T. Chang, P.-C. Hsu, S.-C. Kao, M.-C. Lin, and C.-C. Hsu, Method and Apparatus for Unifying Self-Test with Scan-Test during Prototype Debug and Production Test, European Patent No. 1,364,436, May 24, 2006.

R7.4 RTL Design for Testability

[Aktouf 2000] C. Aktouf, H. Fleury, and C. Robach, Inserting scan at the behavioral level, *IEEE Design & Test of Computers*, 17(3), pp. 34–42, July 2000.

[Ghosh 2001] I. Ghosh and M. Fujita, Automatic test pattern generation for functional register-transfer level circuits using assignment decision diagrams, *IEEE Trans. on Computer-Aided Design*, 20 (3), pp. 402–415, March 2001.

[Huang 2001] Y. Huang, C. C. Tsai, N. Mukherjee, O. Samoan, W.-T. Cheng, and S. M. Reddy, On RTL Scan Design, in *Proc. IEEE Int. Test Conf.*, pp. 728–737, November 2001.

[IEEE 1463-2001] IEEE Std. 1463-2001, IEEE Standard Description Language Based on the Verilog Hardware Description Language, IEEE Press, New York, 2001.

[Ravi 2001] S. Ravi and N. Jha, Fast test generation for circuits with RTL and gate-level views, in *Proc. IEEE Int. Test Conf.*, pp. 1068–1077, November 2001.

[Roy 2000] S. Roy, G. Guner, and K.-T. Cheng, Efficient test mode selection and insertion for RTL-BIST, in *Proc. IEEE Int. Test Conf.*, pp. 263–272, October 2000.

[Wang 2005b] L.-T. Wang, A. Kifli, F.-S. Hsu, S.-C. Kao, X. Wen, S.-H. Lin, and H.-P. Wang, Computer-Aided Design System to Automate Scan Synthesis at Register-Transfer Level Test, U.S. Patent No. 6,957,403, October 18, 2005.

[Zhang 2003] L. Zhang, I. Ghosh, and M. S. Hsiao, Efficient sequential ATPG for functional RTL circuits, in *Proc. IEEE Int. Test Conf.*, pp. 290–298, October 2003.

R7.5 Concluding Remarks

[IEEE 1149.1-2001] IEEE Std. 1149.1-2001, *IEEE Standard Test Access Port and Boundary Scan Architecture*, IEEE Press, New York, 2001.

[IEEE 1149.6-2003] IEEE Std. 1149.6-2003, *IEEE Standard for Boundary Scan Testing of Advanced Digital Networks*, IEEE Press, New York, 2003.

[IEEE 1500-2005] IEEE Std. 1500-2005, *IEEE Standard for Embedded Core Test*, IEEE Press, New York, 2005.